Intelligent Assistants for Flexibility Management
(Grant Agreement No. 957670)

D6.7 Final Application-specific iFLEX Assistant prototypes

Date: 2024-03-04

Version 1.0

Published by the iFLEX Consortium

Dissemination Level: PU - Public

Co-funded by the European Union’s Horizon 2020 Framework Programme for Research and Innovation under Grant Agreement No 957670

Document control page

Document file: D6.7 Final Application-specific iFLEX Assistant prototypes.docx
Document version: 1.0
Document owner: **ECE**

Work package: WP6 System integration and service packaging
Deliverable type: DEM - Demonstrator, pilot, prototype

Document status: ☒ Approved by the document owner for internal review
☒ Approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Andraž Javernik (ECE)	21.12.2023	ToC
0.1	Andraž Javernik (ECE)	19.01.2024	Updated section HEMS in Slovenian pilot
0.2	Nikos Charitos (ICOM)	01.02.2024	Updated sections on UI & A&M Interface for the Greek & Slovenian pilots, and DRMS for the Greek pilot.
0.3	Jussi Kiljander (VTT)	07.02.2024	Finnish pilot updates
0.4	Marion Paraschi, Vivi Dalakou (HERON)	12.02.2024	Greek pilot contribution
0.5	Dušan Gabrijelčič (IJS)	19.02.2024	Slovenian pilot RAI, DT, TMM, SPI, DR section update
0.6	Jussi Kiljander (VTT)	22.02.2024	Slovenian pilot and Greek pilot AFM updates
1.0	Andraž Javernik (ECE), Sara Žerjav (ECE)	22.02.2024	Executive summary, introduction, conclusion

Internal review history:

Reviewed by	Date	Summary of comments
Christos T. Krasopoulos	2024-02-29	Accepted with minor comments and suggestions for modifications
Athanasios Papakonstantinou	2024-03-03	Approved with minor comments

Index:

1 Executive summary

The goal of task 6.7, "Final Application-specific iFLEX Assistant prototypes," is to deploy the iFLEX Assistant (iFA) application to end-users. The third piloting phase of the iFA software framework enhances the efficiency of the system, which can manage demand and production flexibility for consumers and prosumers. The framework is based on data provided by the Home Energy Management System (HEMS), Building Energy Management System (BEMS), Distributor System Operator (DSO), and energy supplier. Task 6.7 is responsible for integrating the developed iFA components with local platforms and implementing the necessary communication protocols and metering devices. The deployment of the application is made easier with the use of containerization technology.

The third piloting phase version of the iFLEX framework consists of different functional components, most of which have been finalized and deployed: Resident interface, Facility Manager Interface, Resource Abstraction Interface (RAI), Automated Flexibility Management (AFM), Digital Twin Repository (DTR), Weather Service Interface (WSI), Aggregator & Market Interface Module, Demand Response Management Systems (DRMS), Real-Time Energy Metering & Actuation Platform (REMAP), Enrolment Module, Home Energy Management System (HEMS), User Interface (UI at house and building levels), Smart Metering, and Weather Data ingest Module.

The development of the common iFLEX framework is a continuous and iterative process executed with the collaboration of pilot users in all phases. In this third piloting phase, the iFA application is deployed to end-users after determining that it supports all relevant functionalities and meets the necessary requirements.

The focus is also on deploying, testing, and evaluating the installed HEMS hardware to provide further details regarding the functionalities and limitations of the hardware components used in this phase. An illustration of connecting an integrated HEMS\BMS with external interfaces is provided, showcasing the setup's complexity and functionality. These hardware components facilitate the measurement and control of various devices within the iFLEX project at users' sites, including heat pumps, domestic hot water heaters, solar power plant inverters, distribution electricity meters, other electricity meters and district heating.

2 Introduction

2.1 Purpose, context and scope

This deliverable focuses on the Finnish, Greek and Slovenian third piloting phase which covers deployment and implementation of higher-level hardware and software functional components and assessing the technical functionality of the iFLEX framework primarily in relation to data collection, flexibility control, predictions, and user web browser (testing mode) or mobile app base interface (for user data visualization and HEMS, BEMS system manipulation). The deliverable also describes the updates that have been made to the functional components in order to form and deploy application-specific iFLEX Assistant prototypes in the Finnish, Greek and Slovenian pilot.

2.2 Content and structure

This document is structured in six sections and the focus is on section 3 as follows:

- Section 3, describes in detail the evolution of iFLEX framework prototypes during the third piloting phase, in which they were deployed to the end-users and integrated with the local aggregation platforms. Deployment view is illustrated for each pilot country, including the descriptions of iFA end-user interfaces, and the collaboration between iFA and existing solutions of local prosumers.

2.3 Key changes in this deliverable according to the previous one

The overall scope of changes in the third piloting phase according to first and second piloting phase can be summarized as follows:

Finnish pilot:
The iFLEX Assistant applications descriptions have been updated. The main change is the inclusion of a new iFA deployed for a supermarket provided by the MakingCity project (see D8.4 and D8.5 for further information about the cooperation with MakingCity). In the third pilot phase, both iFA deployments (supermarket and apartment building) include all iFA components and are integrated with Enerim's aggregation platform that is connected to Nord Pool intraday test market.

Greek pilot:
The description of the application-specific iFLEX Assistant prototype for the Greek pilot has been updated according to the specifications of the third pilot phase. More specifically, the deployment diagram that presents the iFLEX components and external systems, which will be integrated into the pilot by the end of this phase, as well as the communication protocols between them have been modified. Furthermore, the functionalities of the following components or systems are elaborated:

- User Interface component,
- Aggregator and Market Interface component,
- ICOM's Demand Response Management System,
- Heron's Real-time Energy Metering & Actuation Platform,
- HERON Smart Monitoring mobile application.

Slovenian pilot:
The third piloting phase deliverable has been updated with information on designed, developed, integrated and deployed components forming together an application of the specific iFLEX assistant prototype used in the Slovenian pilot. In particular, the following components have been updated based on the advances in the last piloting period:

- HEMS deployment and installation, together with needed details for application integration,
- Illustration of RAI deployment and application integration,
- Automated Flexibility Management,
- Digital Twin deployment and functionality,
- User Interface's instantiation for the Slovenian pilot and supported features,
- Aggregator and Market Interface's functionalities.

3 IFLEX Assistant prototype

3.1 Finnish iFLEX Assistant prototype

3.1.1 Overview

The focus in the third phase Finnish pilot is on deploying the iFLEX Assistant (iFA) in a new building (supermarket) and demonstrating local flexibility management and aggregation for Nord Pool intraday market. The Finnish pilot consists of two iFAs. The first iFA is deployed for the apartment building (already part of phase 1 and phase 2) and the second for the new supermarket integrated via cooperation with the MakingCity project.

The apartment building iFA is responsible for managing energy and flexibility for the whole apartment building. Please refer to the *HLUC-3: Manage flexibility at building community level*, documented in D2.1 [1] for further details on the use case. This use case covers an important consumer sector in Finland, because roughly 40% of people[1] live in an apartment building and it is typical that the majority of the energy costs in an apartment building (e.g., heating, warm water, sauna) are shared by the housing cooperative.

The iFA is deployed into an apartment building with 90 apartments. The apartment building has a Building Management System (BMS) that monitors and controls the building's heating ventilation and air conditioning (HVAC), as well as, lighting, electric sauna and elevators. Figure 1 presents the stakeholders and external systems of the apartment building iFA.

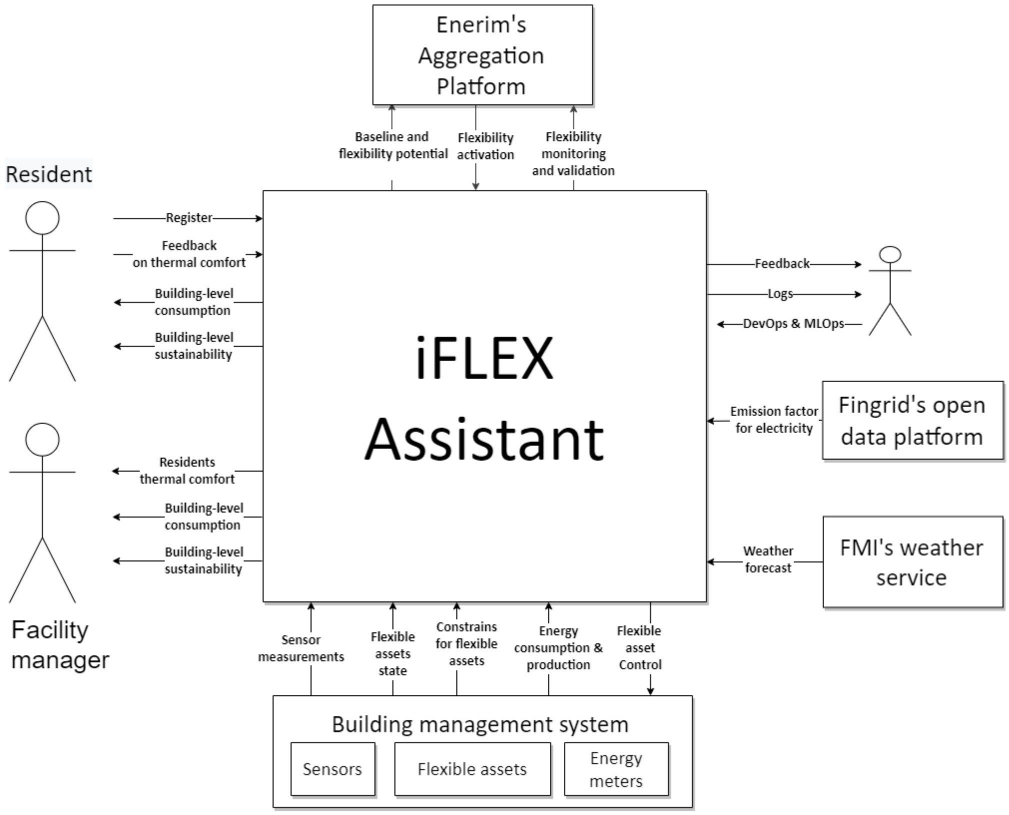

Figure 1: iFLEX Assistant of the apartment building pilot with relevant stakeholders and external systems

There are two types of end-users: Residents and Facility manager. Please refer to iFLEX deliverables D2.1 [1] and D2.3 [2] for further details on the services provided to these end-users.

The iFA interacts with several external systems. The BMS provides iFA with means to monitor and control building assets. The apartment building had an old BMS system without any external interfaces. For this reason, a JACE-8000 gateway was deployed into the building. The iFA accesses the following measurements and controls the hybrid heating system via the JACE-8000:

[1] https://www.stat.fi/til/asas/2017/asas_2017_2018-05-17_tie_001_en.html

- Building level electricity consumption, power and frequency at one-minute sampling rate.
- District heating energy consumption at one-minute sampling rate.
- Ventilation units' return air temperature and optionally return air relative humidity and CO_2.
- Average indoor air temperature, relative humidity, and CO_2 of the building.
- Various control-related parameters, including set-point and measured values for water temperatures across the system (e.g., space heating, domestic hot water, exhaust air heat pump supply), as well as, status information (on water pumps', fans', control valves and heat pump compressor status).

Fingrid Open Data platform provides the iFA with estimates of CO_2 emissions for electricity consumptions. Finnish Meteorological Institute (FMI) provides the weather forecasts utilized by iFA in forecasting the energy consumption and flexibility.

The context view of the iFA deployment for the supermarket is almost identical to the apartment building depicted in Figure 1. The main difference is that there are no residents and thus no information exchange related to residents. The information exchanged via the BMS is also slightly different. In the supermarket the iFA exchanges the following information with the building automation system:

- Energy metering from the whole building and a variety of submetering points, including HVAC-R system and PV panels.
- A variety of process parameters from the HVAC-R system, including e.g. CO_2 (refrigerant) and water temperatures, mass flow (CO_2 and water), valve positions.
- Indoor temperature measurements from three sensors inside the building and three ventilation machines.
- Control commands (sent by the iFA) for adjusting the space heating water temperatures, and heating of air in individual ventilation machines.

Both of the buildings are connected to Enerim's Aggregation Platform. Enerim's Aggregation Platform is connected to the Nord Pool intraday test market to demonstrate and validate explicit DR in the energy wholesale market.

3.1.2 Deployment view

In the third phase, the iFLEX Assistant deployments (apartment building and supermarket) consist of the following functional components: Resident Interface (only deployed in the apartment building), Facility Manager Interface, Resource Abstraction Interface (RAI), Automated Flexibility Management (AFM), Digital Twin Repository (DTR), and Weather Service Interface (WSI). These logical components are implemented with concrete software artifacts as follows:

- **Resident Interface:** The resident interface consists of Resident frontend, Resident backend and User Database (DB) artifacts. The Resident frontend is implemented as web browser application, usable with mobile phones and computers. The Resident backend is responsible for interfacing with the RAI component to fetch relevant data for the Resident interface. It is implemented with Node.js. The User DB is implemented on top of MongoDB. These back-end software artifacts are deployed on a Resident server.
- **Facility Manager Interface:** The facility manager interface consists of a variety of dashboards to visualize building energy parameters. It is implemented on top of Grafana framework[2]. The Grafana back-end is hosted on the same server as the software artifacts of the RAI module.
- **Resource Abstraction Interface:** The third pilot phase implementation of the RAI components consists of following software artifacts: oBIX database, Building Energy Management System (BEMS) interface for apartment building, BEMS interface for supermarket, Fingrid interface, and Client interface. The oBIX database stores all the data collected from BEMS, Fingrid, Finnish Meteorological Institute (FMI). The AtmosCare BEMS interface is implemented with Java and provides a mechanism for collecting and controlling building resources. It interfaces with a JACE gateway of the BEMS system with standard oBIX protocol. The supermarket BEMS interface is implemented with Python. It provides interfaces for collecting measurement data and sending control commands to the supermarket's BEMS via the MakingCity Oulu ICT Platform [3]. The Fingrid interface component is implemented with Java and provides means to collect CO_2 emission data from Finnish Transmission System Operator

[2] https://grafana.com/

(TSO), Fingrid. The Client interface provides other functional components of the iFLEX Assistant access to the data stored in oBIX database. The RAI components are deployed in a single server, called RAI server.

- **Digital Twin Repository:** The digital twin repository of the iFLEX Framework consists of a variety of Digital Twins (DTs) for the consumer, including people and buildings they live in. There are separate Digital Twins deployed for the apartment building and the supermarket. Both Digital Twins are implemented with Python programming language on top of scipy[3] and tensorflow[4]. The apartment building DT consists of models for estimating the building's baseline load profiles both for electricity and district heating, as well as, flexibility model, including indoor temperature model, heat pump model and space heating model. The Digital Twins is implemented as a single software artefact (Python class) that is interfaced by the AFM component.
- **Automated Flexibility Management:** The AFM component is realized as a single process that imports also the Building's twin provided by the DTR component. Separate AFM instances are deployed for the apartment building and the supermarket. The AFM is implemented with Python programming language, and it provides an MQTT interface for flexibility management. In the third phase the AFM interface will be demonstrated and tested via a specific interface. In the third phase, the AFM interface will be integrated to the Aggregator and market interface component.
- **Weather Service Interface:** The WSI component deployed for the Finnish pilot interfaces with the Finnish Meteorological Institute (FMI) weather forecast service and stores the forecast into the oBIX database. The FMI interface is implemented as a single process that is deployed into the same server as the RAI components.
- **Aggregation and Market Interface:** The Aggregation and Market Interface (A&M Interface) provides an MQTT interface for Enerim's Aggregation Platform to interface with the iFLEX Assistant (i.e., it provides messages on the baseline and flexibility forecasts and enables activation of the flexibility provided by the consumer). Additionally, the A&M Interface provides the AFM module with means to fetch energy price information from Enerim's Aggregation Platform.

Figure 2 illustrates the deployment view of the iFLEX Assistant deployed in the apartment building. Again the deployment in the supermarket is almost identical with the exception that there is no End-user interface and associated back-end services for the residents.

[3] https://www.scipy.org/
[4] https://www.tensorflow.org/

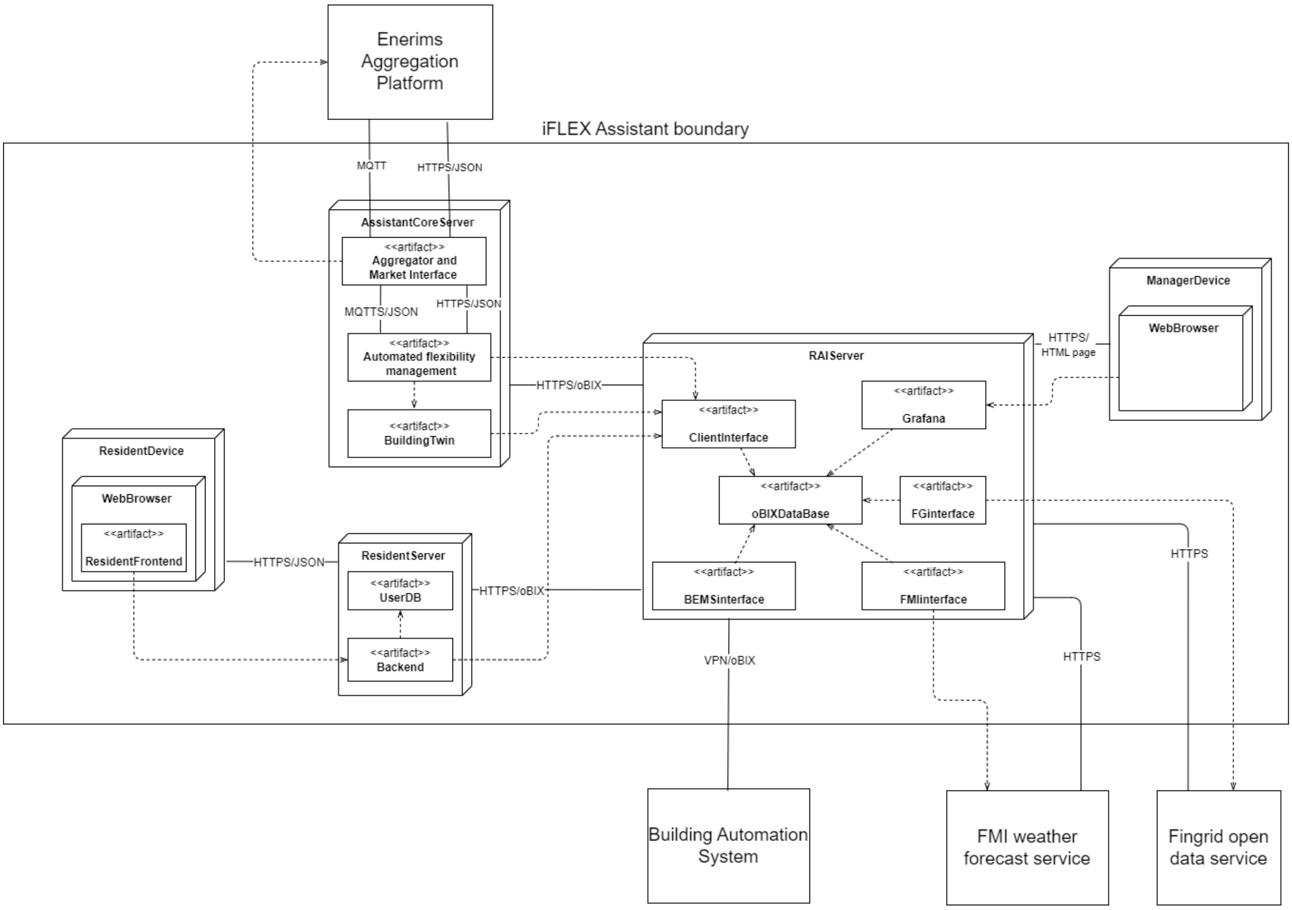

Figure 2: Deployment view of the iFLEX Assistant prototype for the apartment building iFLEX Assistant.

3.1.3 iFLEX assistant in the containerized environment

The core parts of the iFA are deployed in cloud environment as a containerized application. These parts of iFA include AFM, required models from digital twin repository and interfaces. All of these components have been developed using Python version 3.9.5. Pip package management system have been used in the development phase in order to share identical environment among the developers. Following packages are required by the iFA and therefore installed to Docker environment:

```
pandas==1.5.2
paho-mqtt==1.6.1
requests==2.28.2
joblib==1.2.0
matplotlib==3.6.2
scikit-learn==1.2.0
holidays
PyYAML==6.0
protobuf==3.20.3
tqdm==2.2.3
tensorflow-cpu==2.14.0
keras==2.14
lightgbm==4.1.0
```

Figure 3: Required packages in Conda environment

Using containerization technology, iFA can easily be deployed into the cloud platform and this technology enables creating new iFA instances with minimal human effort. In iFLEX, Docker and Docker-compose are used as a containerization technology.

The following workflow demonstrates, how iFA instances can be created for the pilot. Docker builds container images using dockerfile as a source of instructions and Figure 4 shows instructions to create an iFA instance.

```
# syntax=docker/dockerfile:1
FROM python:3.9-slim
WORKDIR /instances/
COPY requirements_ema.txt .
COPY /instances/ /instances/

COPY /ema /ema
ARG PIP_NO_CACHE_DIR=1
ARG GIT_COMMIT=unspecified
LABEL git_commit=$GIT_COMMIT
RUN apt-get update && apt-get install -y \
    curl \
    git \
    python3-pip \
    && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
RUN pip install -r requirements_ema.txt
CMD ["python", "main.py", "--conf-file", "online_cfg.yml"]
```

Figure 4: Dockerfile instructions for iFA

First, it copies the requirement file presented in Figure 3 to image and copies also required source codes. Docker image can be created in X86-based machine in Linux/Mac terminal with command "*docker build --tag building_ema -f Dockerfile --no-cache .*" In the case of an ARM-computer used as a development machine, the following command is used "*docker-buildx build --platform linux/amd64 --tag ema:latest --build-arg GIT_COMMIT=$(git log -1 --format=%h) -f Dockerfile_ema --no-cache --load .*" With that command, we can cross-compile Docker images to be used in X86 servers

When Docker images are created, containers need to be created based on the images. Docker Compose is used to create and manage multi-container applications. It uses docker-compose.yml configuration file to define how applications are created based on Docker images and how applications interact with each other.

Figure 5 describes how an AFM component is created based on a Docker image created with dockerfile shown in Figure 4. Docker-compose file also includes a container for visualizing and activating flexibilities and a container that is responsible for managing the forecast history. Using docker-compose, new instances can be easily created just by adding a new service to docker-compose file and modifying the configuration files.

```
version: "3"
services:
    household_afm:
    container_name: household_afm_instance
    image: household_afm
    working_dir: /instances/greek/
    command: python main.py --conf-file config/online_cfg.yml
    volumes:
      - ./household_afm/greek/config:/instances/greek/config
      - ./household_afm/greek/tmp:/instances/greek/tmp
    restart: always
  ema_kaukovainio:
    container_name: kaukovainio_instance
    image: kaukovainio
    working_dir: /instances/kaukovainio/
    command: python main.py --conf-file config/online_cfg.yaml
    volumes:
      - ./ema/instances/kaukovainio/config:/instances/kaukovainio/config
      - ./ema/instances/kaukovainio/tmp:/instances/kaukovainio/tmp
    restart: always
  ema_hoas:
    container_name: hoas_instance
    image: ema
    working_dir: /instances/hoas/
    command: python main.py --conf-file config/online_cfg.yaml
    volumes:
      - ./ema/instances/hoas/config:/instances/hoas/config
      - ./ema/instances/hoas/tmp:/instances/hoas/tmp
      -
./ema/instances/hoas/models:/instances/hoas/trained_models/modelling
    restart: always
  mqtt_visualizer_hoas:
    container_name: mqtt_visualizer_instance_hoas
    image: mqtt_visualizer
    command: bokeh serve --allow-websocket-origin=* mqtt_visualizer.py --
args --conf-file config/visualizer.yml
    volumes:
    - ./visualizer/config:/scripts/config
    - ./visualizer/history:/scripts/history
    - ./visualizer/simulation_hist:/scripts/simulation_hist
    restart: always
    ports:
    - 5006:5006
```

Figure 5: Example docker-compose description

3.2 Greek iFLEX Assistant prototype

3.2.1 Deployment View

The version of the iFLEX Assistant (iFA) for the third pilot phase in the Greek pilot consists of the following iFA components: User Interface (UI), Aggregator and Market Interface (A&M), Resource Abstraction Interface (RAI), Digital Twin (DT), Automated Flexibility Management (AFM), Security and Privacy. Furthermore, it interacts with the following external components: ICOM's Demand Response Management System (DRMS), HERON's Real-time Energy Metering & Actuation Platform (REMAP), a Push Notification Server, a Weather Service, HERON's tariff API and a CO_2 emissions Service.

The following Figure 6 presents the deployment diagram of the iFA's instance for the Greek pilot, whilst a presentation of the different solutions can be found in the following sections.

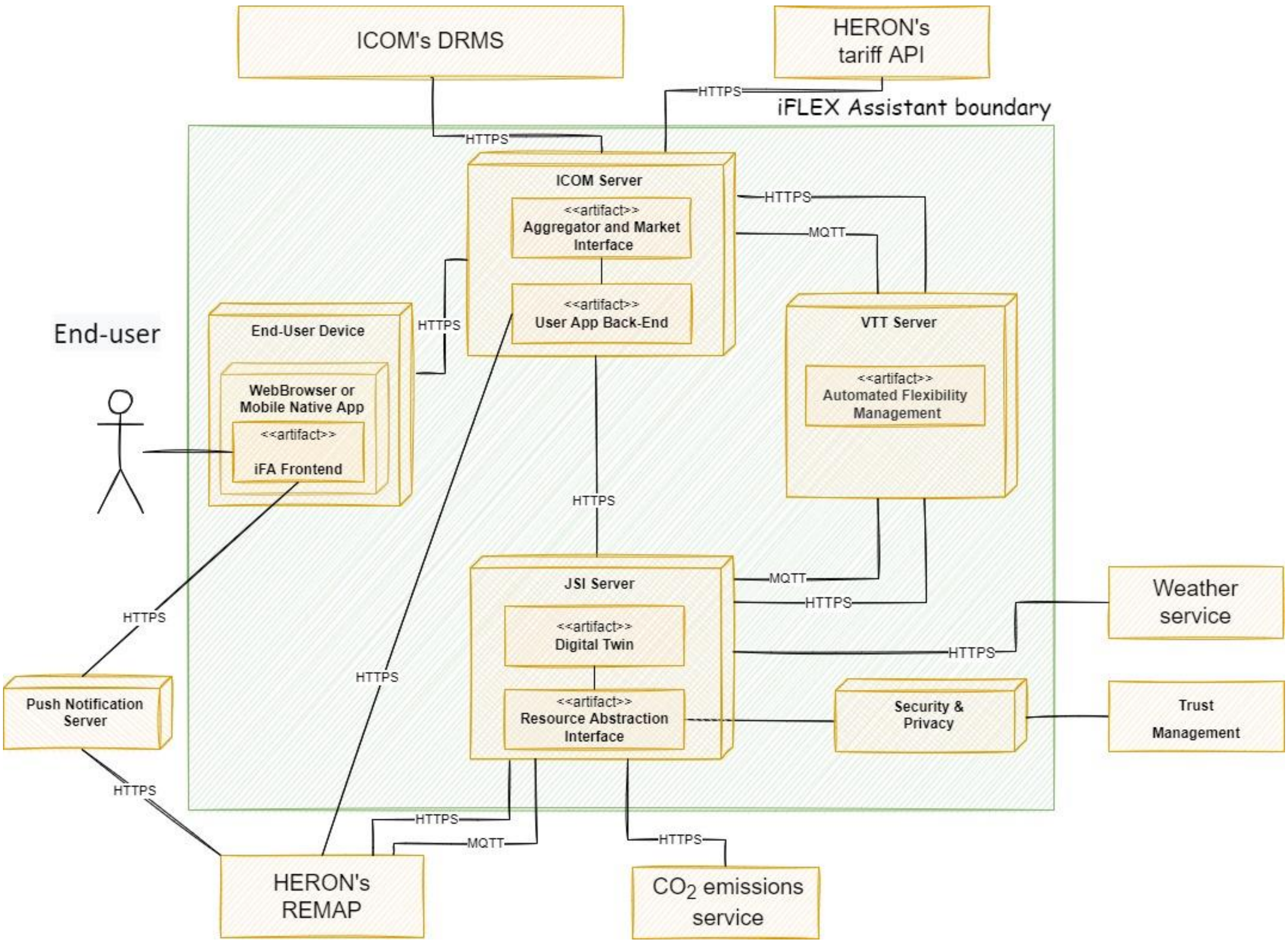

Figure 6: Deployment view of iFA for Phase 3 of the Greek pilot

3.2.2 User Interface App

The user interface module (iFA FrontEnd component in Figure 6) in the Greek pilot will be instantiated from a collaboration between HERON's baseline native application and iFA-specific interfaces developed by ICOM in the project, which will be exposed as a web view (through the User App Back-End in Figure 6). The former shall provide core operations related to User/Home/Device management and energy monitoring (specific to customer offering of HERON), as well as communication of push notifications and security mechanisms, which will be integrated with ICOM's solution. The latter should provide to the pilot users a set of functionalities tailored to the iFLEX project. These functionalities concern user preferences, energy monitoring, Demand

Response (DR) services, visualisation of cost-related information, energy advice, as well as sustainability and energy efficiency performance tracking.

More specifically, the following requirements related to iFA's UI component [4] for the Greek pilot should be supported for the third piloting phase:

- FN-UI-01 – Operation mode customization
- FN-UI-02 – User-defined time and operational constraints
- FN-UI-03 – End-user feedback
- FN-UI-04 – Optimisation policy selection
- FN-UI-05 – Automation level customization
- FN-UI-08 – Provision of consent for the schedules of dispatchable assets
- FN-UI-09 – DR notification policy
- FN-UI-10 – Insights into sustainability metrics
- FN-UI-11 – Real-time energy data
- FN-UI-12 – Past energy data
- FN-UI-13 – DR reports
- FN-UI-14 – Insights into energy efficiency
- FN-UI-15 – Customised alerts
- FN-UI-16 – Energy advising service
- FN-UI-17 – Inspection of energy tariffs
- FN-UI-21 – DR event notification
- FN-UI-22 – Presentation of DR event history
- FN-UI-26 – View asset schedules when offline
- FN-UI-27 – Actual schedules of assets
- FN-UI-28 – Estimation of energy costs
- FN-UI-29 – Benefits from iFA's operation
- NF-UI-01 – Integration of the iFA's UI for the Greek pilot into HERON's app

3.2.3 Aggregator & Market Interface Module

The A&M Interface module will exchange signals with ICOM's Demand Response Management System (DRMS) based on OpenADR2.0 protocol [5]. The DR-related sub-module is based on the client side (i.e., Virtual End Node – VEN) of OpenLEADR[5], an open-source implementation in Python[6], compliant with OpenADR. When needed, extensions to the OpenADR protocol are employed, so that the DR-related requirements of the iFLEX project can be fully met. The A&M Interface component is also responsible for retrieving the electricity tariffs from a dedicated API exposed by HERON. It also transforms external DR- and price-related signals to the iFLEX data model, so that they can be communicated to the other components of the iFA. During this phase, functionalities related to both DR and energy tariffs are supported.

More specifically, the features below should be supported by the A&M Interface Module [6] of the Greek pilot for the third piloting phase:

- FN-AM-04 – Information on participation in explicit DR actions
- FN-AM-05 – Communication of flexibility potential
- FN-AM-06 – Access to DR reports to end users
- FN-AM-07 – Accessing flexibility information history from external system
- FN-AM-08 – Receiving flexibility signal
- FN-AM-09 – Communication of flexibility signal
- FN-AM-10 – Response to flexibility signal (explicit DR)
- FN-AM-13 – Communication of electricity tariffs from external system
- FN-AM-14 – Access to electricity tariffs to end users
- FN-AM-17 – Access of flexibility validation data from the RAI (explicit DR)
- FN-AM-18 – Communication of flexibility validation data (explicit DR)
- FN-AM-19 – Assessment of end user participation in a DR event (explicit DR)
- FN-AM-20 – Communication of participation assessment data (explicit DR)

[5] https://github.com/openleadr
[6] https://www.python.org/

3.2.4 ICOM Demand Response Management System

The solution of ICOM's DRMS, which will be deployed in the Greek pilot, should support the following functionalities [6]:

- FN-DR-01 – iFA end-users' flexibility potential
- FN-DR-03 – Sending Flexibility Signal
- FN-DR-04 – Response to Flexibility Signal (explicit DR)
- FN-DR-05 – Information on participation in explicit DR actions
- FN-DR-06 – Access of flexibility validation data from the iFA (explicit DR)
- FN-DR-08 – Response to flexibility request
- FN-DR-09 – Flexibility dispatch
- FN-DR-10 – Provide activated flexibility report
- FN-DR-11 – Calculate individual performance and participation rate metrics of DR participants
- FN-DR-12 – Calculate statistics per DR event and DR program

The DRMS employs the OpenADR2.0 protocol to communicate with iFA, while extensions to this protocol are utilised to meet iFLEX requirements when necessary. The DR communications solution is based on the OpenLEADR implementation, as in the case of iFA's A&M Interface module, but on the server side (i.e., Virtual Top Node – VTN). As regards the interaction with the BRP system, which is operated by OPTIMUS in the context of the Greek pilot, flexibility requests are exposed through a REST API, documented via OpenAPI specification[7].

3.2.5 Automated Flexibility Management

The AFM component functions as a unified process, incorporating the household twin from the Digital Twin Repository. The Energy Planner part of the AFM is identical to the Finnish pilot. A new controller was implemented for boiler control as described in D3.9 [7]. The AFM supports following functional requirements:

- FN-AFM-01 - Provide baseline forecasts
- FN-AFM-02 - Flexibility potential
- FN-AFM-03 - Activate offered flexibility
- FN-AFM-06 - Provide schedule information for the End-user Interface

3.2.6 HERON's REMAP

HERON operates an integrated platform consisting of software and hardware systems for energy monitoring and management, the Real-time Energy Metering & Actuation Platform (REMAP). REMAP communicates with smart meters, smart plugs and other IoT devices based on the MQTT protocol. The data is stored in InfluxDB databases and can be parsed through an API to iFLEX RAI providing an interface for iFLEX Assistant in order to access energy data and provide asset control commands to the on-premises assets. In more detail, the metering infrastructure utilized includes real-time power meters, a back-end system for measurements collection and storage and a remote API to provide access to iFA. Furthermore, following the Revised Greek pilot specifications outlined in Deliverable D7.2 [8], HERON's platform has been updated to accommodate the inclusion of additional smart assets such as various types of smart plugs. A Grafana dashboard (Figure 7) which can be accessed by pilot participants independently of the iFLEX app has been provided.

[7] https://swagger.io/specification/

Figure 7: Smart Plugs Grafana dashboard

3.2.7 HERON Smart Monitoring mobile application

HERON has developed a mobile application that utilizes REMAP as its backend in collaboration with iFA. The application (Figure 8) allows users to view data of their consumption and home status data in both real time and past time (historical data) and interacts with iFA allowing them to schedule the use and remote control their electrical water boiler.

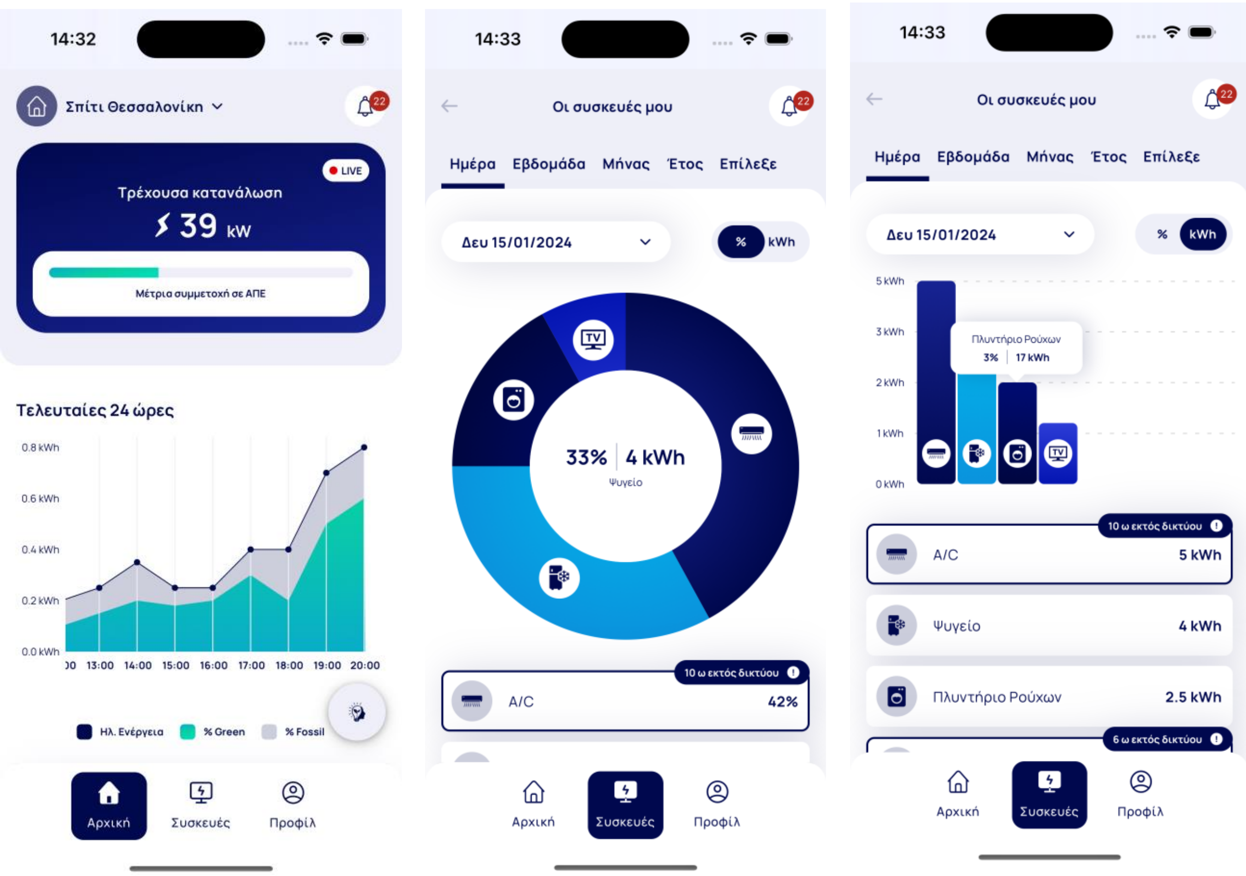

Figure 8 : HERON mobile application with iFA integration

3.3 Slovenian iFLEX Assistant prototype

The Slovenian iFLEX Assistant framework prototype in the third piloting phase consists of the following functional components: an Enrolment Module, Home Energy Management System (HEMS), Resource Abstraction Interface (RAI), Smart Metering and Weather Data Ingest Module, Digital Twin, Automated Flexibility Management Module, Demand Response Module, Trust Management Module, Security and Privacy Interface and End-User Interface.

The third phase Slovenian pilot focuses, as has been defined in deliverable D7.3 [9], in terms of deployment and integration, the focus was on deploying, testing, and evaluating the Home Energy Management System (HEMS) in real-world conditions. This involved working with a pilot group of users up to and including the third phase, alongside the necessary pilot infrastructure and background components. Furthermore, integration validation was conducted for the input data sources with the background system. Efforts were also directed towards enhancing the design and implementation of the iFLEX Assistant end-user interface. The status of this work is reported in Section 3.3.1 on deployment view. In this section, at first, the status of the components and iFLEX Assistant subsystems are briefly abstracted. The components' deployment illustration beyond the reports given so far is then reported in the section subsections.

According to the plan specified in D7.3 [9], the iFLEX Assistant prototype will be used in the third piloting phase with a focus on implementation of primary use cases and pilot deployment. The targeted use cases to be addressed were selected and described in the Final architectural deliverable D2.5 [10].

3.3.1 Deployment view

At the beginning of the third phase piloting, the iFLEX Assistant framework consists of a number of components:

- **Enrolment module**: the Enrolment module enables user friendly and privacy compliant enrolment of the end users in the pilots. The users get invited in the project by the pilot host through the module. The module is used to enable pilot users to give consent for data collection and processing and to initialise basic data structures needed for backend system operation. More information on the enrolment procedure is available in the deliverable D4.8 [11]. The enrolment module can be accessed at *uporabnik.iflex-project.eu*. The name 'uporabnik' stands for the word "user" in Slovenian. The reason for such a name is because the module is a frontend for all current and future piloting users activities.
- **HEMS**: the HEMS gets installed and connected with the rest of the framework during the enrolment procedure. It is an essential component for sensing, data collection and activation of devices and their flexibility in pilot user's households. HEMS functionality and piloting deployment are further described in Section 3.3.3,
- **Resource Abstraction Interface**: the Resource Abstraction Interface (RAI) presents a component that collects data from HEMS and other external sources, and stores and provides it to other iFLEX components, such as the Enrolment Module (EM), Digital Twin (DT) and End User Interface (EUI). The RAI provides interfaces for actuators available at HEMS as well. RAI design and implementation have been reported in the deliverable D4.3 [12]. Third piloting phase RAI has been implemented as an extension of a Prosumer Cloud Service (PCS). The following functionality has been deployed:
    - Services for data collection from the HEMS third phase implementation are based on MQTT protocol. The data include measurements from sensors, devices configuration information and devices state. The service receives designed and implemented MQTT messages from HEMS devices periodically, on One-minute intervals, and stores it in the PCS storage,
    - Services for data ingest from external services an interface has been provided for ingest of data from external data sources like smart meter data and weather data. The data is stored in the PCS and is aggregated in other time slices (1h, 2h, 6h, 1d, 1w, etc.),
    - Collects weather data for selected household locations from Open Weather Map (OWM) and stores the data in the PCS storage,
    - Provides backend service for selected data collected in enrolment process, needed for DT operation,
    - Provides backend service for storing DT models data as well as household building parameters. The RAI will be likely used for storing some model states in the next iteration,

  - Exports all the selected data to the rest of components in an as unified way as possible via designed and implemented Resource Abstraction Interface (RAI) API. Aggregates of the data are available for many collected data, some will be added in the next iteration,
  - Provides information on tariffs and prices per household. The interface will be moved to Aggregation & Marketing interface in the next iteration,
  - Provides control functionality over assets that can be controlled by the project. The assets include on-off devices, Photo Voltaic (PV) installations, heat pump set-points, etc.
- **External data ingest module**: The ingest module pushes the data from the data provider towards the RAI module. The ingest module has two parts, the first being the pusher and the second the ingest. The deployment of both modules is as follows:
  - The pusher transfers data from the source towards the ingest module. In the case of Slovenian pilot the pusher pushes smart metering data from Elektro Celje (ELE) data sources towards ingest module. The deployment is based on existing software services available at ELE; the same services are used to provide data to other ELE clients, like transmission operator, retailers, etc.
  - The ingest provides an FTP server for receiving the pusher data. The ingest reads the inputs periodically, transforms the inputs according to the ingest specification requirements and pushes the data towards the RAI module. The RAI implementation stores the data in the backend storage.
- **Digital Twin**: The digital twin module implements models as discussed in the deliverable D3.3 [13]. The module is currently integrated close to the RAI module, since the data access is crucial to model operation. The deployed module provides the following functionality:
  - The heat and heat consumption demand physical models,
  - Heat pump COP model,
  - Scheduling interface for control from Automated Flexibility Module,
  - REST interfaces to control the DT, access the data calculated by the twin and to load and store models from the RAI storage,
  - Automated scheduling and test collection interface to perform automated tests and experiments on the households.
- **Automated Flexibility Management:** The AFM component functions as a unified process, incorporating the household twin from the Digital Twin Repository. The Energy Planner and Controller part are identical to the Finnish pilot described in the section 3.1.1.
- **Trust Management Module**: The Trust Management Module (TMM) enables cryptographic trust management between system entities like system components, end users, etc. The TMM is designed to support two types of trust management: X.509 Public Key Infrastructure and Self Sovereign Identity (SSI) based trust management. TMM module design and implementation is further reported in D4.8 [11].The X.509 management is fully implemented, SSI one is in an experimental stage. The TMM has been integrated with the EUI and AFM.
- **Security and Privacy Interface**: The Security and Privacy Interface (SPI) provides access control functionality based on security and privacy policies, identities or roles of the entities in the system established through secure communication with iFLEX framework services and enforcement mechanisms provided by the services. The SPI utilizes identities and their credentials as these are created and managed through Trust Management Module. The SPI module is further described in D4.8 [11]. Both REST and MQTT interfaces are supported. The SPI module is already deployed and controls access to the RAI and DT module.
- **Demand Response**: The updated Demand Response management solution is based on the combination of SCOM's background solution and open-source implementation of OpenADR specification, the OpenLEADR[8]. The updated DRMS prototype is capable of emitting and receiving flexibility events information as well as viewing and managing events via a web interface. While the first phase was focused on development of initial DRMS prototype and its integration with the iFA for

[8] See OpenLEADR home page for details: https://www.lfenergy.org/projects/openleadr/

the purposes of collecting relevant data from various heterogenous sources, the second phase expands this work by exploiting the updated DRMS for carrying out experiments with execution of flexibility events. For this purpose, the user's flexibility potential is provided by iFA to the DRMS, which then dispatches the flexibility events (signals) according to the potential and event schedule. Relevant information on the executed events is fed back to the DRMS, which processes the information and delivers the Flexibility report to the iFA. Some additional information on Demand Response Management System and Marketing & Aggregation module is described in Sections 3.3.5 and 3.3.7 respectively.

- **End-User Interface**: The End-User Interface enables the interaction of the end users with the iFLEX Assistant. In the third phase, the user has ability to set their preferences and assist them in managing both energy and flexibility. The UI was designed with the intention to be as simple as possible, while at the same time providing the users with useful insights. The EUI is described in more details in Section 3.3.88.

3.3.2 Advanced Metering System

An Advanced Metering System (AMS), or Advanced Metering Infrastructure (AMI) is understood as a system of smart meters, communication infrastructure and IT systems that enables measurement, remote reading and management of data on the use of electricity for the purposes of billing, monitoring the quality of supply and operation and planning of distribution networks. The emphasis is on providing sufficiently frequent information about usage, billing according to actual consumption, using advanced tariff systems, implementing consumption management measures, and developing and implementing new services.

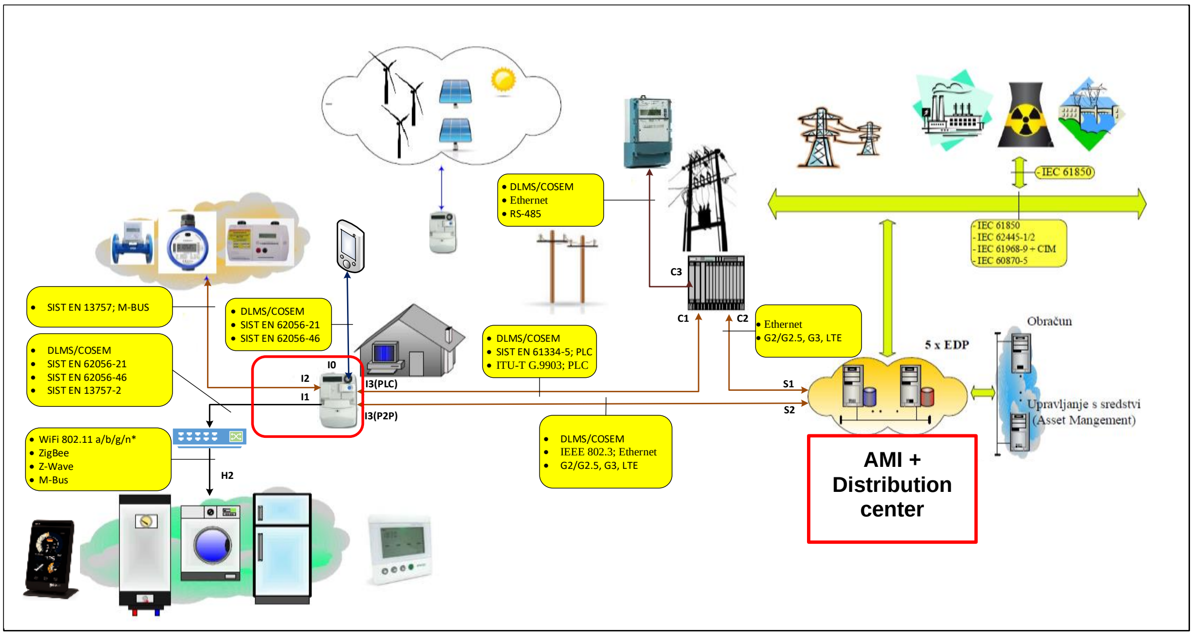

Figure 9: Building an advanced metering system

Interfaces on the smart meter:

- I0 local service interface for DSO,
- I1 interface intended for system users for local access to data and information (dedicated display, Smart Home systems, HEMS, BEMS, etc.),
- I2 interface for local connection with other meters (gas, heat, water...),
- I3 interface between the smart meter and the DSO (PLC OFDM G3 or 4G(LTE)).

Sending data to the I1 interface, as we do for active users in the iFLEX project is for local access to data for the purposes of displaying data, transmitting measurement data to other modules, home devices and systems like HEMS or BEMS.

Basic technical requirements for this interface are:

- one-way communication channel intended exclusively for reading sent data compliant with SIST EN 62056-7-5;
- HAN channel (one-way communication from the meter to house systems and devices such as: Energy House Displays (IHD), Smart House Systems (SM) and Home Energy Management System (HEMS);
- speed ≥ 2400 b/s, basic setting 2400 b/s;
- receiving data requires protection using appropriate cryptographic encryption methods as described in the DLMS/COSEM standard (Green Book, Edition 7 and Edition 8).

The main goal of introducing an advanced metering system is to provide system users with better information and services, and to enable them to better manage their consumption. The configuration of PUSH intervals and objects that are sent to I1 is configurable via the I0 and I3 interfaces. The default configuration for the test samples is specified in Table 1 and Table 2.

Table 1: Data sent from the I1 interface every 5 seconds

Parameter number	Parameter name
0-0:42.0.0	COSEM name
0-0:96.1.2	Device ID3
1-0:32.7.0	voltage L1
1-0:52.7.0	voltage L2 only 3 phase meter
1-0:72.7.0	voltage L3 only 3 phase meter
1-0:31.7.0	current L1
1-0:51.7.0	current L2 only 3 phase meter
1-0:71.7.0	current L3 only 3 phase meter
1-0:1.7.0	A+ current active power
1-0:2.7.0	A- current active power
1-0:3.7.0	R+ current reactive power
1-0:4.7.0	R- current reactive power

Table 2: Data sent from the I1 interface every 1 hour

Parameter number	Parameter name
0-0:42.0.0	COSEM name
0-0:96.1.3	Device ID4
1-0:1.8.1	Energy A+ T1
1-0:1.8.2	Energy A+ T2
1-0:2.8.1	Energy A-T1
1-0:2.8.2	Energy A-T2
1-0:3.8.1	Energy R+ T1
1-0:3.8.2	Energy R+ T2
1-0:4.8.1	Energy R-T1
1-0:4.8.2	Energy R-T2

The I3 communication interface between the meter and the DSO (WAN) is intended for two-way communication with the HES in AMI.

The technical requirements for this interface are:

- PLC communication
  - Narrow-band OFDM G3-PLC (G3-PLC Alliance),
  - G3 PLC chip: frequency range CENELEC A and FCC;

- RF communications (the communication module must simultaneously support 4G and 2G mobile networks)
  - 4G (LTE) mobile network with frequencies of 800, 900 and 1800 MHz
  - 2G mobile network (fallback) 900 and 1800 MHz

3.3.3 HEMS in Slovenian pilot

In the third pilot stage of the iFLEX project, changes took place in the Slovenian pilot, pertaining to the technical part of the HEMS hardware installed at residential end users. Hardware by another manufacturer was updated. Examples of equipment installation at pilot users will be presented here in after.

3.3.3.1 HEMS functionalities and limitations

In the third Slovenian piloting phase the following hardware functional components are used:

- HEMS main control unit:
  - Supports Wi-Fi and Ethernet communication with Resource Abstraction Interface (RAI) and user devices in the Local Area Network (LAN) or via external expansion models connected within the LAN.
  - MODBUS TCP/RTU communication with user devices.
  - Enables power supply via an external 12 VDC connection (power supply unit) or PoE (Power over Ethernet).
  - At the beginning of the third pilot phase, when industrial users started to get involved in the project, it was found that communication via the GSM/UMTS communication protocol was needed, which was then ensured by a hardware upgrade on the HEMS controller
- Energy Management System (EMS) to Ethernet gateway:
  - Supports wired communication with heat pumps by the manufacturer BOSCH via an EMS bus.
  - The interface can be powered via EMS bus, Ethernet (PoE) or an external 12 VDC power supply unit.
  - The interface connects to an internal LAN by wired or wireless connection.
  - The interface communicates with the main HEMS control via MQTT or REST API data exchange protocol. In the third pilot phase, a new plug-in was developed for communication via the EMS bus with the BOSCH heat pump, which communicates with the main unit via the Modbus RTU communication protocol
- M-bus/P1 TTL to Ethernet gateway:
  - Supports wired communication with smart electricity distribution meters with lower nominal power ratings (< 50 kW) by the manufacturer Landis+Gyr and Iskraemeco.
  - The interface can be powered by wire via Ethernet (PoE) or from a 12 VDC power supply unit.
  - The interface connects to the LAN via Ethernet (wired connection) or wirelessly via a Wi-Fi communication interface.
  - The interface communicates with the main HEMS control via UART protocol over RS485 logical levels.
- RS-485 to Ethernet gateway:
  - It supports wired communication via MODBUS RTU protocol with the devices that support MODBUS communication (heat pumps, car chargers, solar power plant inverters, industrial distribution meters for electricity with power capacity of over 50 kW).
  - The interface can be powered by wire via Ethernet (PoE) or from a 12 VDC power supply unit.
  - The interface connects to the LAN by Ethernet (wired connection) or wirelessly via a Wi-Fi communication interface.
  - The interface communicates with the main HEMS control via MQTT or REST API data exchange protocol.
- Smart plugs to Ethernet gateway:
  - Smart plugs allow electricity consumption measurement and ON/OFF control for user devices that do not support other digital communication protocols that would allow monitoring their consumption and controlling them.
  - The smart plug is powered from 230 VAC power mains at the intended spot of user device control.
  - A smart plug connects to the LAN wirelessly via a Wi-Fi network.

- The smart plug communicates with the main HEMS control via MQTT or REST API data exchange protocol.

- Iskraemeco P1 to WiFi gateway
    - In the third pilot phase, there was a need for a wireless connection between the smart distribution meter and the main HEMS controller. The reason for this is the considerable distance of the smart distribution meters from the actual installation of the HEMS controller and the disabled cable access to the P1 port on the distribution meter.
    - The mentioned gateway enables a wired connection to the P1 port, from which it is powered and at the same time collects measurement data, which is then forwarded via a wireless connection within the local LAN network where the main HEMS controller is installed.
    - The external P1 to WiFi gaeway and the main HEMS controller further exchange measurement data via wireless WiFi connection and REST API protocol.

The figure below illustrates an example of connecting an integrated HEMS complete with external interfaces (Figure 10).

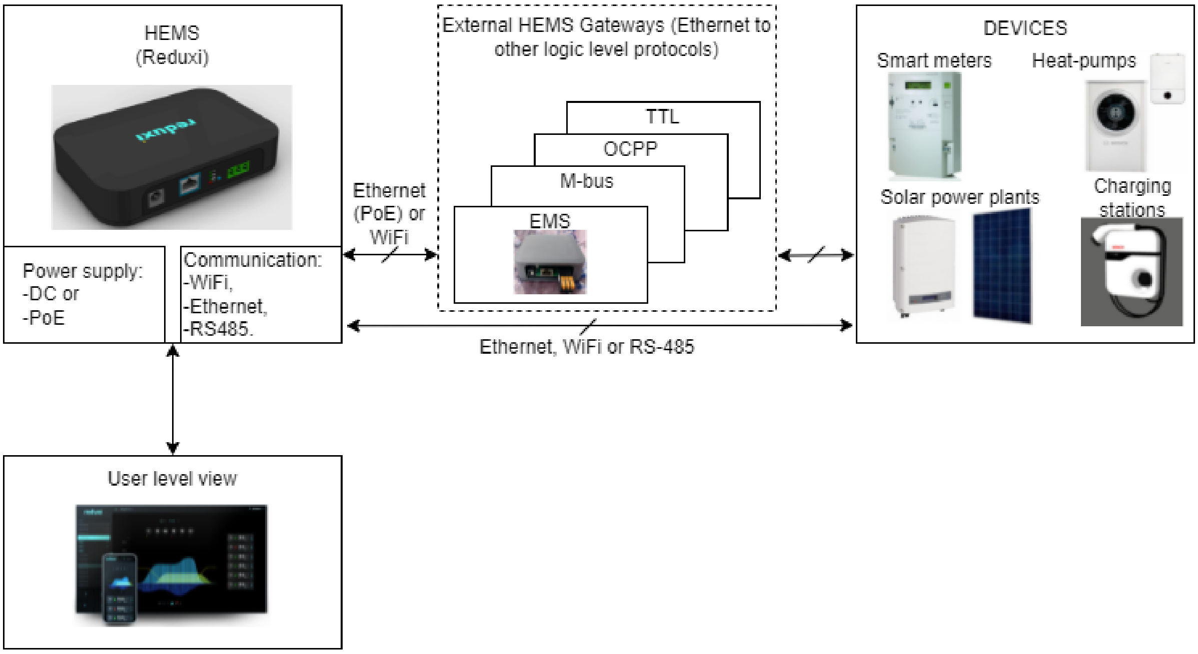

Figure 10: Connection of the HEMS with end user devices

3.3.3.2 Measurement and control signals from a HEMS device

The external measurement/control units referred to above, which communicate with the main HEMS control unit, can be used to address many devices within the iFLEX project at the user's site. The most commonly addressed devices within the pilot area in Slovenia include the following:

- heat pump,
- domestic hot water heater,
- solar power plant inverter (PV inverter),
- distribution electricity meter,
- other electricity meters.

3.3.3.3 Heat pump

The HEMS device and relevant external measurement/control units allow obtaining measurement data from the heat pump, as well as controlling the heat pump. Measurement/control signals are read or sent to the heat pump via EMS or MODBUS RTU (RS-485) communication protocol.

The following measurement data can be read from the heat pump:

- electric energy consumed,
- electric energy produced,
- coefficient of performance (COP) defined as electric energy consumed / electric energy produced,
- temperature of the supply (outlet/hot water) line of the hot water tank,
- temperature of the return (cold water) line of the hot water tank,
- temperature inside the hot water tank,
- temperature at respective thermostats installed within the premises,
- environment temperature as measured at the external unit of the heat pump.

Write or control signals can also be sent to the heat pump for the following parameters:

- changing the desired temperature at each respective thermostat,
- changing the desired temperature inside the hot water tank,
- switching the heat pump off and on.

In second and third pilot phase the connectivity to the following heat pump manufacturers has been added:

- Kronoterm (Modbus RTU communication protocol),
- BOSCH (communication over EMS room thermostat lines),
- NIBE (Modbus RTU communication protocol),
- TIKI (Modbus RTU communication protocol),
- GREE (Modbus RTU communication protocol),
- MITSUBISHI (Modbus RTU communication protocol).

An example of connection to the Kronoterm heat pump via the RS485 bus is shown in the figure below.

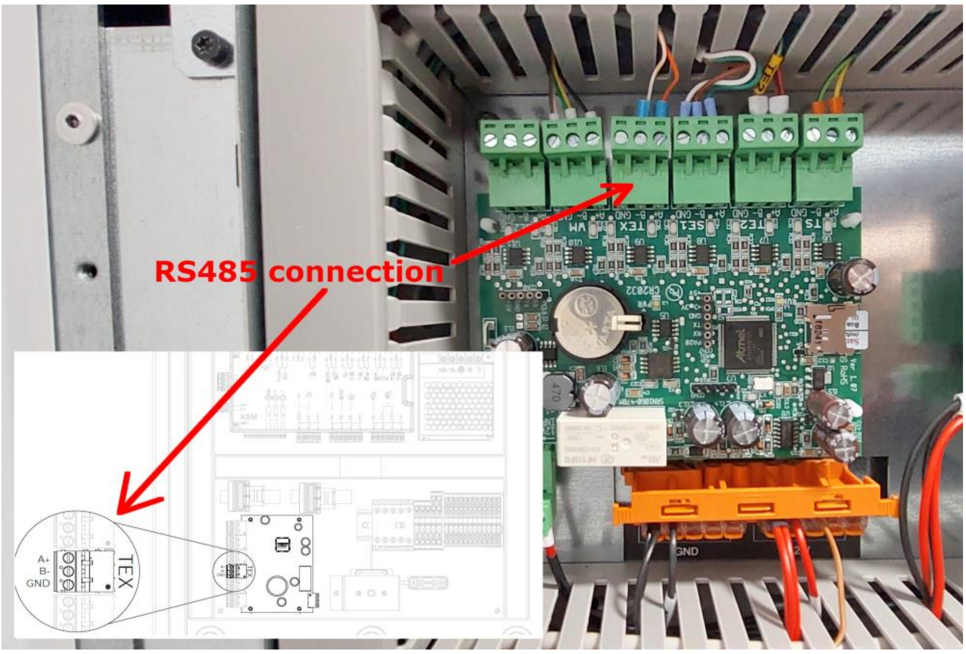

Figure 11: RS485 connection to Kronoterm KSM main unit

3.3.3.4 Domestic hot water heater

Most domestic hot water heaters use a simple thermostat for temperature control. The thermostat senses the temperature of the hot water inside the heater and switches the heater on or off depending on the desired/set temperature. The main HEMS unit connects to the domestic hot water heater via an external smart plug that allows power consumption measurement and control of the heater. The smart plug intended for domestic hot water heater control allows reading the following measurement signals:

- electricity consumption,
- domestic hot water temperature measurement (optionally via a PT100 temperature sensor).

The domestic hot water heater can also be controlled with the ON/OFF mode via a relay switch. Currently, the smart plug allows control of single-phase alternating current (AC) devices with a nominal voltage of 230 VAC and a maximum current of 16 A (which is equivalent to ~3,600 WAC). The image below shows the installation of the PT100 sensor and the smart socket.

Supported connectivity to the next smart plugs:

- Shelly (Communication over REST API).

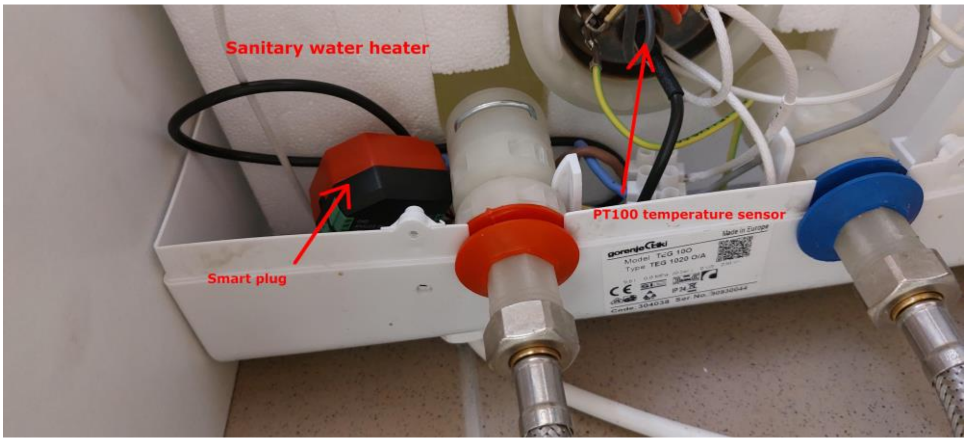

Figure 12: Installation of a smart socket and temperature sensor PT100

3.3.3.5 Solar Power plant Inverter (PVI)

A solar inverter, or photovoltaic inverter, is the component of a solar power plant that converts the Direct Current (DC) output of a photovoltaic solar panel into Alternating Current (AC). In the Slovenian pilot, we address the users with installed self-sufficient supply solar power plants, the power of which is defined by a maximum of 80 % of the residential unit's connected load to the distribution network. For example, if a unit is connected to the distribution network via three 25-amp fuses (3× 25 A), which corresponds to a connected load of 17 kW, then power from the solar plant can be fed back into the network with a maximum of 80 % of connected load, which is 13,6 kW in this particular case. Many users opt for solar power plants that can, at a given moment, feed into the network electric energy with a power that exceeds the limit specified by the power distribution company. At such times, the difference between the maximum allowed power sent to the grid and the available power of the solar power plant is lost. Such loss represents the potential to use the iFLEX project to exploit within the residential unit the energy that would otherwise have been lost. For this purpose, we include solar power plants in the Slovenian part of the iFLEX project, which can be monitored and controlled as necessary via the main HEMS unit and external measurement/control units. The following measurement data can be read from the solar inverter:

- voltage and power on each phase of an alternating connection,
- frequency,
- current solar power plant power,
- electric energy produced.

At a certain point, the inverter can also be controlled to decrease the power of feeding excess electric energy into the grid (in percentages relative to the inverter's nominal power, e.g., 100 % means full power of feeding power into the grid) if it is found that this would improve the network conditions.

Supported connectivity to the next solar power plant inverters:

- SolarEdge (Modbus RTU and Modbus TCP communication),
- Deye (Modbus RTU communication),
- FOX (Modbus RTU communication),
- HUAWEI (Modbus RTU communication),
- KACO (Modbus RTU and Modbus TCP communication),
- SMA (Modbus RTU communication),

- VICTRON ENERGY (Modbus RTU communication).

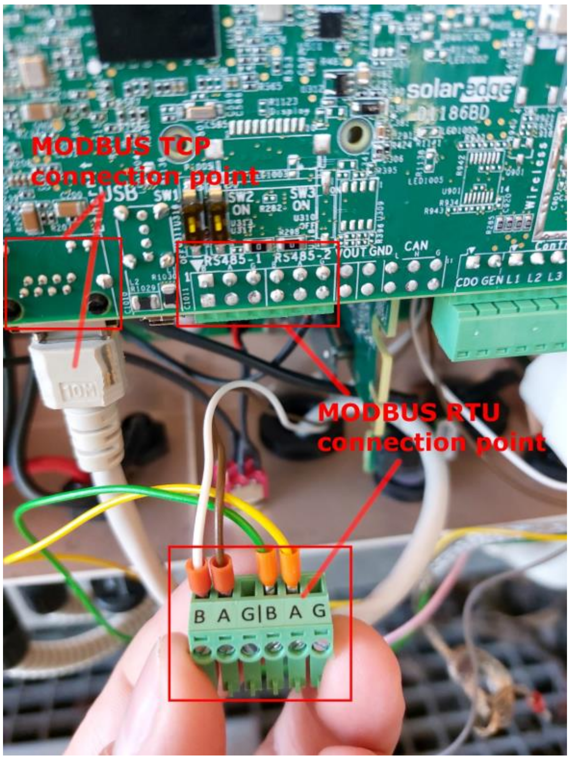

Figure 13: The connection point for the HEMS unit to the self-sufficient supply solar power plant via SolarEdge Modbus RTU communication protocol

3.3.3.6 Distribution electricity meter

Distribution electricity meter is a device installed by the power distribution company at the point where a residential unit is connected to the distribution network (power grid). In Slovenia, electricity meters by the manufacturers Landis+Gyr and Iskraemeco are predominantly used. These meters allow the power distribution company to collect data at 15-minute intervals and transfer them once per day to their information system via GSM communication or via PLC communication (communication via power lines). The meters installed at the unit's connection point allow one-way communication for reading measurement data intended for the user. Readings of measurement data is enabled via M-bus (meter bus) or via RS-485 (MODBUS RTU) communication protocol. The following measurement data can be read from the distribution electricity meter, using the main HEMS unit, at an interval of a minimum of five seconds:

- current and voltage for each phase,
- frequency,
- power of feeding the excess electricity into the grid and consumption,
- reactive and active electric energy,
- date and time.

Supported connectivity to the next DSO meters:

- Iskraemeco (DLMS/COSEM over TTL and RS485 for industrial DSO meters),
- Landis+Gyr (DLMS/COSEM over M-bus and RS485 for industrial DSO meters).

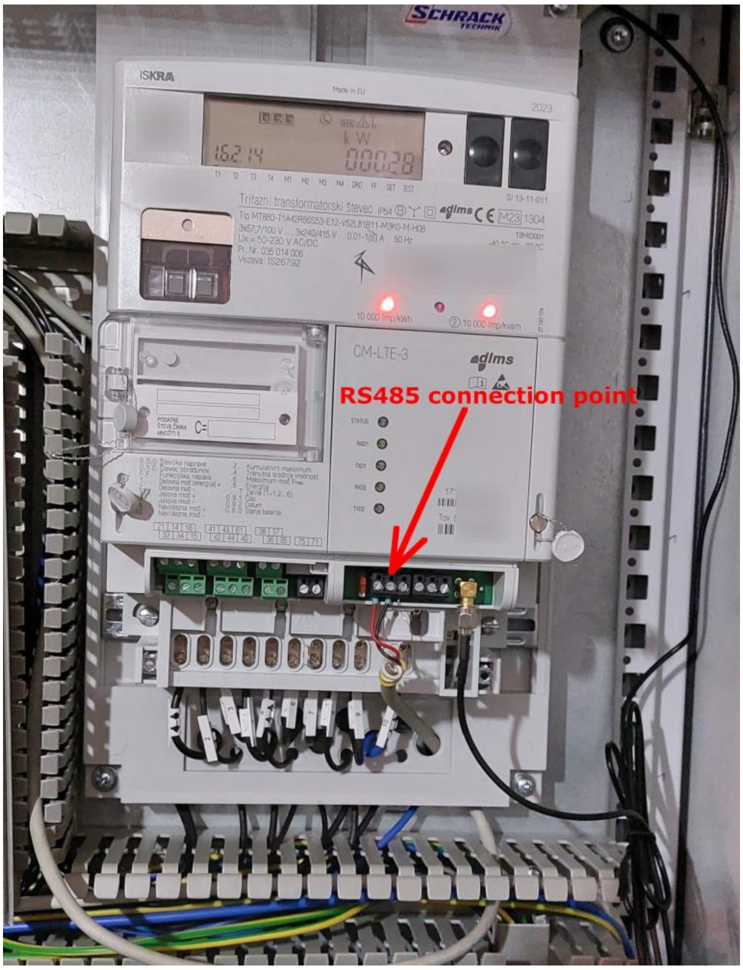

Figure 14: Connection to the industrial DSO electricity meter Iskraemeco via RS485 one-way (half-duplex) and DLMS/COSEM communication protocol

3.3.4 HEMS deployment and integration illustration

The HEMS is integrated with the RAI via MQTT protocol. Three major types of messages are supported, defined in D4.2 [14]: device configuration messages, device state messages and measurement messages. Each message is communicated over its own, household specific topic. The HEMS MQTT part resides at deployed Reduxi HEMS modules, at each household. Example of messages exchanged are given below, using the control tool provided to partners. The tool queries the RAI database first for the household data, then for the devices configuration data. The configuration data is essentially the configuration message received through the MQTT. The configuration data reports on devices, attached to HEMS, their properties, sensors and controls through actuators. The last message is presented as a query for the devices state. Again, the state is as reported trough MQTT. Three messages are denoted with darker grey colour for easier navigation.

```
xyz:~/delo/src/iflex/ece:> python3 control.py -k certs/si.manager@iflex.p12 -e 4JDB-OR78-0J8L-WY39 -t
INFO    2022-12-01 14:44:57,169 control [85]: -> Household 325adfaf-0371-4dad-9ac4-789adab7c0ca info:
INFO    2022-12-01 14:44:57,169 control [86]: {
  "id": "325adfaf-0371-4dad-9ac4-789adab7c0ca",
  "bt": "2022-11-24T13:34:21.661000",
  "connection_type": 3,
  "current_limit": 25,
  "geolocation": {
    "latitude": 46.xyz,
    "longitude": 15.xyz
  },
```

```
  "hems_id": "4JDB-OR78-0J8L-WY39",
  "household_data": null,
  "household_type": "office",
  "p_id": "101",
  "power_limit": 17,
  "tariff_type": "one_tariff"
}
INFO    2022-12-01 14:44:57,185 control [94]: -> HEMS 4JDB-OR78-0J8L-WY39 configuration:
INFO    2022-12-01 14:44:57,185 control [95]: {
  "elements": [
    {
      "device_id": 2,
      "device_name": "ECE kuhinja",
      "device_type": "Actuator",
      "control": [
        {
          "control_type": "on/off",
          "actuator_id": 284,
          "control_name": "Defines on/off control capabilities of a device"
        }
      ],
      "sensors": [
        {
          "sensor_id": 289,
          "sensor_name": "Voltage L1",
          "sensor_unit": "V"
        },
        {
          "sensor_id": 290,
          "sensor_name": "Voltage L2",
          "sensor_unit": "V"
        },
        {
          "sensor_id": 291,
          "sensor_name": "Voltage L3",
          "sensor_unit": "V"
        },
        {
          "sensor_id": 286,
          "sensor_name": "Current L1",
          "sensor_unit": "A"
        },
        {
          "sensor_id": 287,
          "sensor_name": "Current L2",
          "sensor_unit": "A"
        },
        {
          "sensor_id": 288,
          "sensor_name": "Current L3",
          "sensor_unit": "A"
        },
        {
          "sensor_id": 283,
          "sensor_name": "Active power",
          "sensor_unit": "W"
        },
        {
          "sensor_id": 517,
          "sensor_name": "Import - active energy",
          "sensor_unit": "Wh"
        },
        {
```

```
        "sensor_id": 284,
        "sensor_name": "State code",
        "sensor_unit": ""
      },
      {
        "sensor_id": 285,
        "sensor_name": "State name",
        "sensor_unit": ""
      },
      {
        "sensor_id": 282,
        "sensor_name": "Import - active power",
        "sensor_unit": "W"
      },
      {
        "sensor_id": 281,
        "sensor_name": "Communication error",
        "sensor_unit": ""
      }
    ]
  },
  {
    "device_id": 14,
    "device_name": "Razsmernik soncne brez DC napajanja",
    "device_type": "Solar",
    "sensors": [
      {
        "sensor_id": 49,
        "sensor_name": "Current L1",
        "sensor_unit": "A"
      },
      {
        "sensor_id": 50,
        "sensor_name": "Current L2",
        "sensor_unit": "A"
      },
      {
        "sensor_id": 51,
        "sensor_name": "Current L3",
        "sensor_unit": "A"
      },
      {
        "sensor_id": 52,
        "sensor_name": "Voltage L1",
        "sensor_unit": "V"
      },
      {
        "sensor_id": 53,
        "sensor_name": "Voltage L2",
        "sensor_unit": "V"
      },
      {
        "sensor_id": 54,
        "sensor_name": "Voltage L3",
        "sensor_unit": "V"
      },
      {
        "sensor_id": 264,
        "sensor_name": "Active power",
        "sensor_unit": "W"
      },
      {
        "sensor_id": 47,
```

```
          "sensor_name": "Export active energy",
          "sensor_unit": "Wh"
        },
        {
          "sensor_id": 265,
          "sensor_name": "Import active power",
          "sensor_unit": "W"
        },
        {
          "sensor_id": 48,
          "sensor_name": "Export active power",
          "sensor_unit": "W"
        },
        {
          "sensor_id": 142,
          "sensor_name": "Communication error",
          "sensor_unit": ""
        }
      ]
    }
  ],
  "hems_id": "4JDB-OR78-0J8L-WY39",
  "ts": "2022-11-28T19:18:43.523000"
}
INFO    2022-12-01 14:44:57,304 control [106]: -> HEMS 4JDB-OR78-0J8L-WY39 devices state:
INFO    2022-12-01 14:44:57,304 control [107]: {
  "devices_state": [
    {
      "device": 2,
      "state": [
        {
          "bt": "2022-11-28T19:17:41.765000",
          "actuator_id": 284,
          "vs": "off"
        }]}]}
```


The same tool is used below to demonstrate sending of a control message to the household. The message indicates turning of the device 2 actuator 284 on. The message is not sent, -m switch of the tool indicates to generate the message only.

```
xyz:~/delo/src/iflex/ece:> python3 control.py -k certs/si.manager@iflex.p12 -e 4JDB-OR78-0J8L-WY39 -d 2 -a 284 -c on -m
INFO    2022-12-01 14:57:01,391 control [134]: {
  "devices_control": [
    {
      "device": "2",
      "control_device": {
        "actuator_id": "284",
        "command": "on"
      }
    }
  ]
}
```

The measurements are sent to the broker in a form of simple SenML messages as is presented below. The message reports just one measurement in time of the device 3, sensor 160 measurement. The measurement is Import active energy, measured in Wh, with a value of 1,25.

```
{
```

```
  "bn": "urn:dev:3:160",
  "t": 1666775763785,
  "n": "Import active energy",
  "u": "Wh",
  "v": 1.25
}
```


3.3.4.1 RAI deployment illustration

The RAI module in the Slovenian pilot is deployed at JSI premises in a local virtualised cloud environment. The deployment uses Linux LXC container virtualisation for production environment RAI provisioning. The deployment contains both a RAI server as well paired MQTT broker. The server uses Python, MongoDB, Mosquitto, Tornado web server and other tools for the RAI implementation. The services are managed in Linux system start-up scripts style. The Tornado web server is used to provide RAI REST API implementation. At JSI we are working on monitoring solutions for deployed services as well as on developing a Kubernetes environment with proper continuous deployment/continuous integration (CD/CI) functionality for dockerized services.

The RAI interface is used in a programmable manner as is shown below. At first necessary data structures are initialised. Then a Python requests module is used to access the server and retrieve via 'households/<household_id>/building' interface household building parameters. The parameters are used to build a Digital Twin of the household, among the other data.

```
pcs_interface = 'https://si.pcs.e5.ijs.si:8443/iflex/v1/%s'
# Debugging interface
#pcs_interface = 'https://si.pcs.e5.ijs.si:8444/iflex/v1/%s'
ks_file = 'certs/si.manager@iflex.p12'
ks = KeyStore.load(ks_file)
headers = requests.utils.default_headers()
headers['User-Agent'] = 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87
Safari/537.36'

hh_id_one = "1dc43de2-8ac1-4a95-82a1-76bc83e937f1"

url = pcs_interface % 'households/%s/building'
r = requests.get(url % hh_id_one,
            cert=(ks.get_certificate_file(),
                ks.get_privatekey_file()),
            verify=ks.get_ca_certificates_file(),
            headers=headers)
pprint(r.json())

{'air_ventilation': 0.35,
 'bt': '2022-11-30T21:54',
 'floor_area': 213.2,
 'floor_height': 2.4,
 'heatpump_model': 'Bosch Compress 6000 AW 9',
 'hems_id': '64JG-U1PL-UUNQ-ZM23',
 'id': '1dc43de2-8ac1-4a95-82a1-76bc83e937f1',
 'pv_data': {'pv_orientation': 'JZ',
          'pv_power': 15.58,
          'pv_power_limit': 13.6,
          'pv_tilt': 30},
 'temperature_set_point': 22.0,
 'thermal_capacitance_per_floor_area': 165000,
 'u_walls': 2,
 'u_windows': 1.2,
 'ventilation_efficiency': 0,
 'walls_area': 240,
 'weather_location': 'Nova cerkev',
 'window_area_south': 29}
```

In a similar manner the measurement data is obtained and then plotted, as presented in Figure 15. The figure presents power measurements in W for all three phases from a household with installed PV and a heat pump.

3.3.4.2 Digital Twin deployment illustration

The Digital Twin is deployed alongside the RAI module. The DT uses the data directly from the RAI database storage which improves performance over obtaining the data through the REST interface. The twin uses JRC PVGIS[9] for modelling irradiation and household solar gains, the heat demand implementation is initially based on JSI/Comsensus flexibility calculator implementation[10]. The twin provides REST interface for loading and storing Machine Learning (ML) models and for obtaining heat and heat consumption related predictions. An example of a such prediction for the same household as presented in Figure 16.

[9] See PVGIS homepage: https://joint-research-centre.ec.europa.eu/pvgis-photovoltaic-geographical-information-system_en
[10] See the Gitlab repository for details: https://gitlab.com/comsensus/flexibilitycalculator

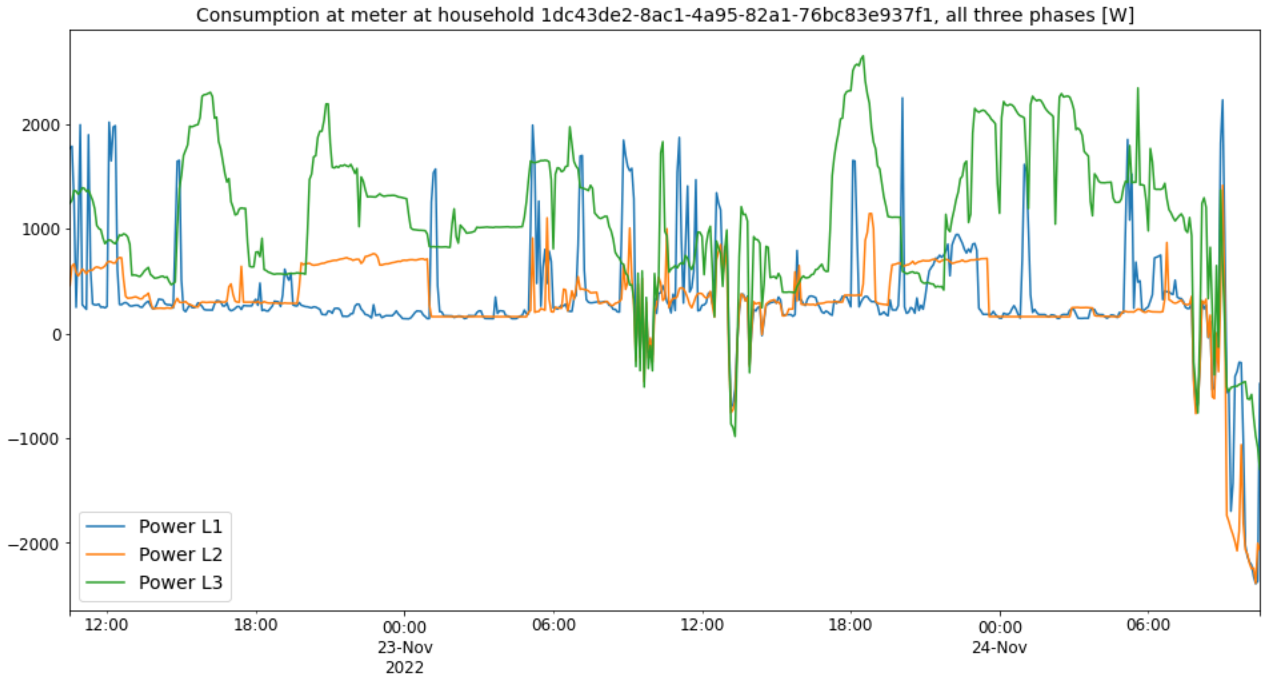

Figure 15: Example of plotted measurement data obtained from a household meter

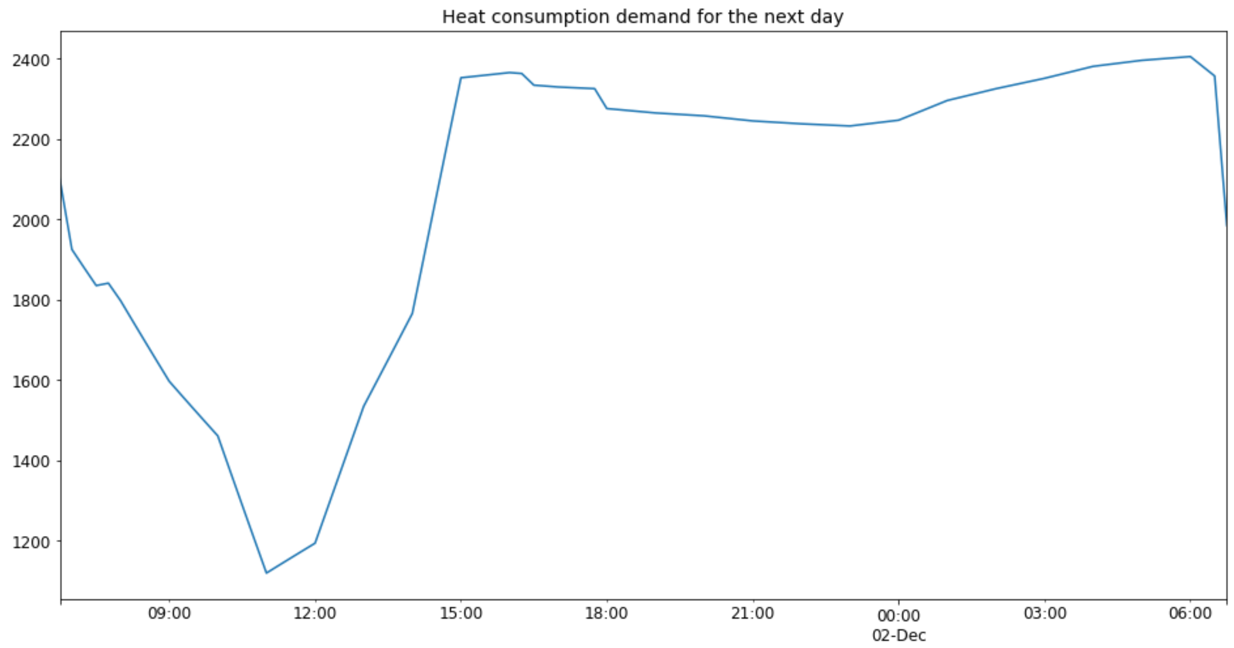

Figure 16: Heat consumption demand for the household with id 1dc43de2-8ac1-4a95-82a1-76bc83e937f1

Modelled COP parameters for the BOSCH Compress 6000 AW 9 are stored in the DT storage and can be used to assess heat pump COP characteristics when the calculations for heat consumption demand are prepared. The Bosch COP model can be obtained from DT API, as shown below.

```
url = rai_interface % 'households/digital_twin/models'
qparams = {'type': 'COP'}
m2url = url + '?' + urllib.parse.urlencode(qparams)
r = requests.get(m2url,
            cert=(ks.get_certificate_file(),
                ks.get_privatekey_file()),
            verify=ks.get_ca_certificates_file(),
            headers=headers)
print(r.ok)
model_data = r.json()
```

The URL in its query part asked for all models of type COP.

```
True
https://si.pcs.e5.ijs.si:8443/iflex/v1/households/digital_twin/models?type=COP
```

Each model is defined with the following keys:

- device_type: heat pump or air-conditioning
- model_type: specific type/model of the device
- type: type of the model
- bt: base time of model creation
- model: the model itself

```
bosch_model = model_data['models'][0]
print(bosch_model.keys())
dict_keys(['device_type', 'model_type', 'type', 'bt', 'model'])
```

The model itself, or only part of the model is presented in the code output below. The model is prepared per temperature degree granulation. At each temperature, based on original graphic characteristics, a polynomial third degree model is calculated.

```
cop_model = bosch_model['model']
print(cop_model)

{'25': {'intercept': 4.776564176564178,
  'coefficients': [0.14886792220125553,
   0.001364746364746358,
   -8.265561598894914e-05]},
```

The COP diagram based on obtained model for four temperatures, namely at 30, 40, 50 and 60 degrees Celsius is shown in Figure 17.

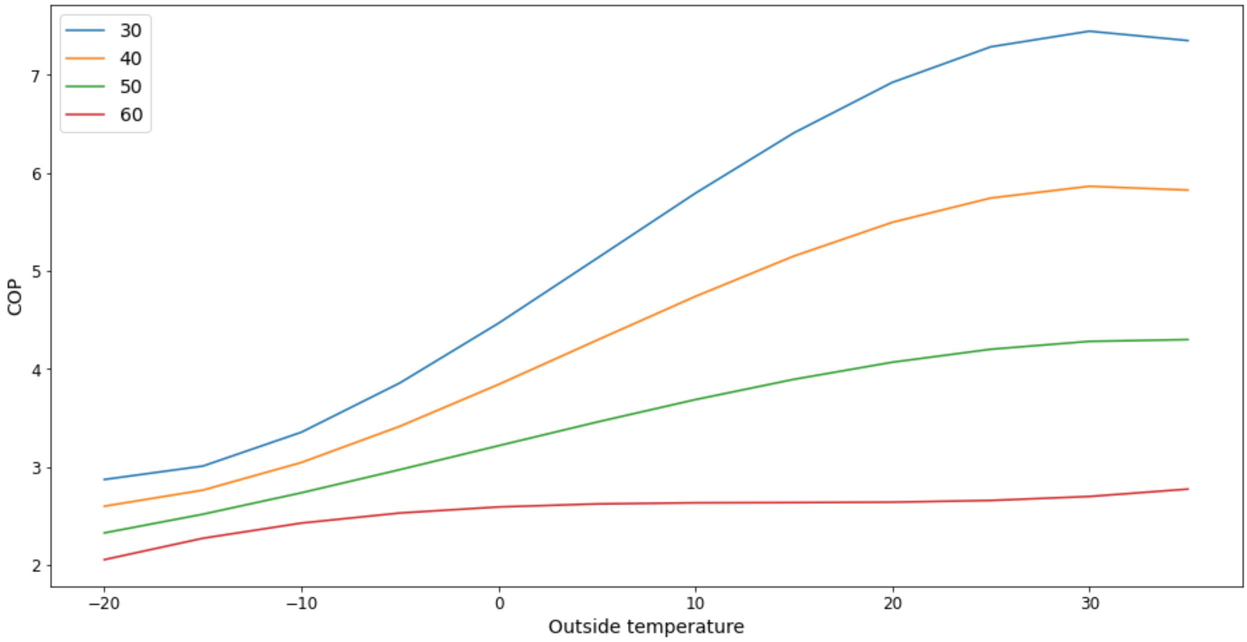

Figure 17: BOSCH Compress 6000 AW 9 heat pump COP parameters model. The curves present characteristics at different inlet water temperature

3.3.5 Automated Flexibility Management

The AFM component functions as a unified process, incorporating the household twin from the Digital Twin Repository. The Energy Planner and Controller parts of the AFM are identical to the Finnish pilot. The AFM supports following functional requirements:

- FN-AFM-01 - Provide baseline forecasts
- FN-AFM-02 - Flexibility potential
- FN-AFM-03 - Activate offered flexibility

Figure 18 - Figure 20 illustrates the baseline, flexibility and DR activation data provided by an AFM instance deployed for a household in the Slovenian pilot.

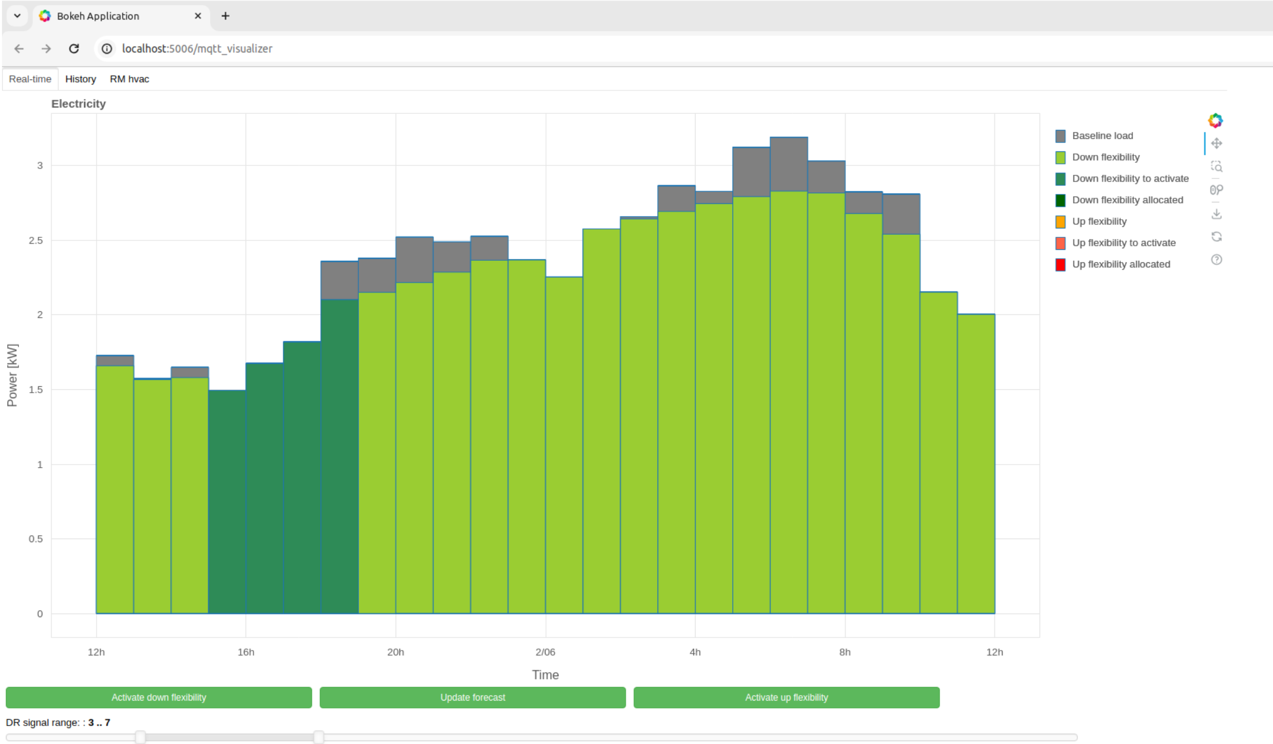

Figure 18: Visualisation of the baseline and flexibility potential provided by the AFM.

The period that would be activated if the DR signal is sent is represented in darker green.

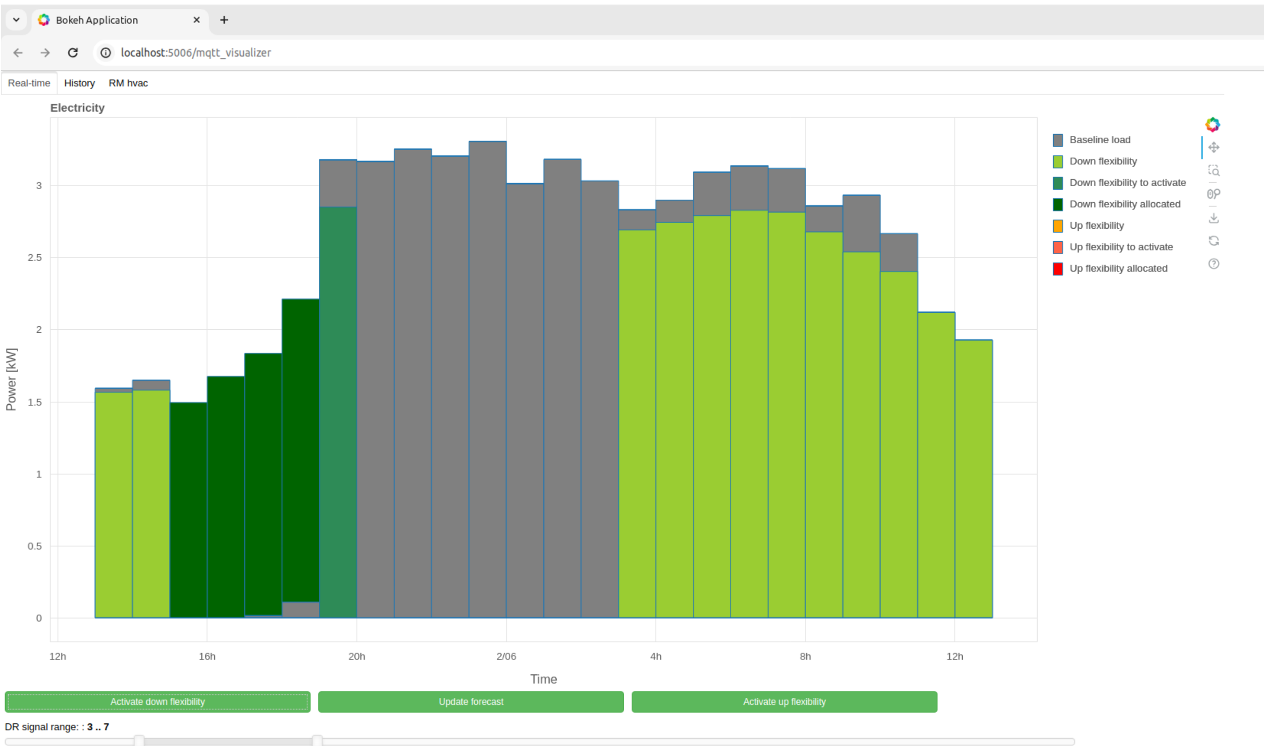

Figure 19: Visualisation of the forecasted baseline and flexibility after flexibility request has been received by the AFM

Flexibility period to be activated is represented in dark green. Notice the “rebound” effect after the DR event in which the energy is paid pack according to the law of energy conversion.

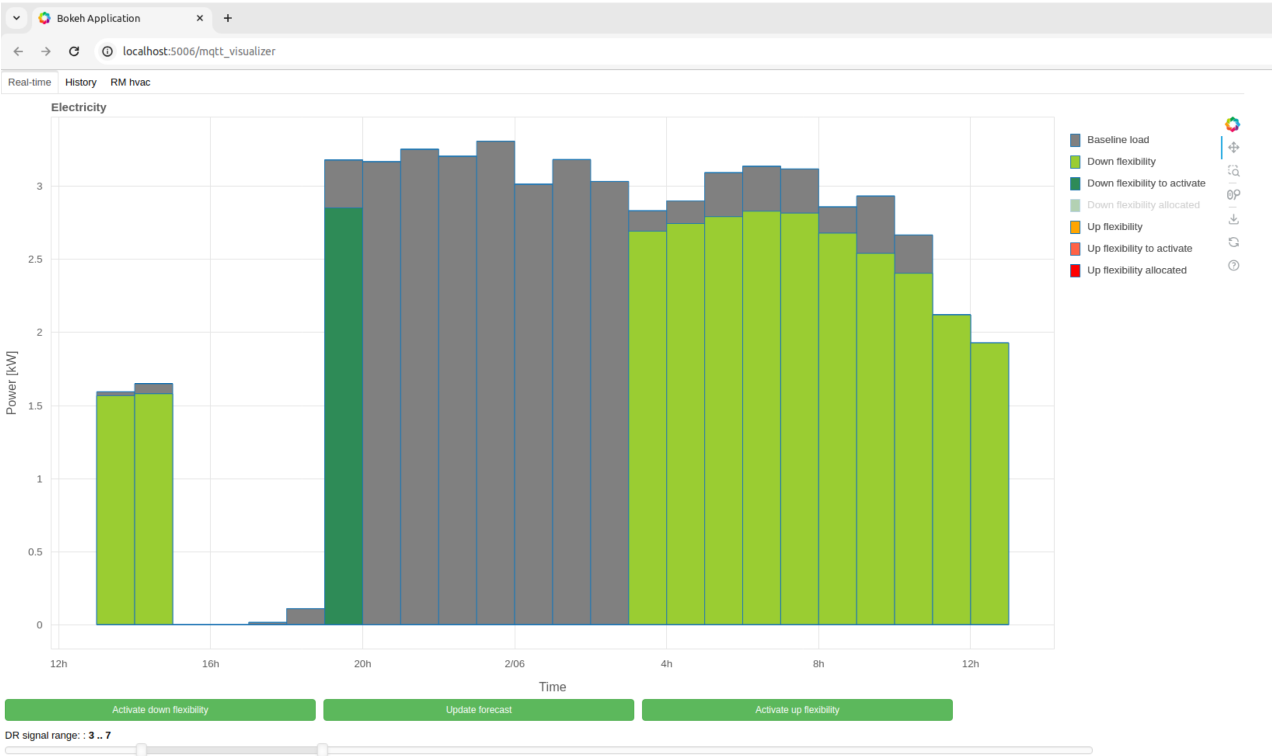

Figure 20: Visualisation of the forecasted baseline and flexibility after flexibility request has been received by the AFM

Flexibility period to be activated is removed to better visualise the forecasted load after the DR event. I.e., this figure contains the same information as Figure 19, but the flexibility period planned to be activated is removed from the visualisations.

3.3.6 SCOM Demand Response Management System

SCOM's DRMS prototype will be deployed and the focus of the phase is on piloting the following functionalities in a scaled pilot:

- FN-DR-01 – iFA end-users' flexibility potential
- FN-DR-03 – Sending Flexibility Signal
- FN-DR-05 – Information on participation in explicit DR actions
- FN-DR-09 – Flexibility dispatch
- FN-DR-10 – Provide activated flexibility report

The solution combines SCOM's preexisting DRMS system, whose backend has been integrated with the OpenLEADR, an open-source implementation of the OpenADR2.0 protocol, to communicate with the Aggregation and Market interface component of the iFA. The DRMS prototype encompasses the OpenADR server side (VTN – Virtual Top Node), which is complemented by its client-side counterpart (VEN – Virtual End Node) provided by the Aggregator & Market interface, which is presented in the next section.

3.3.7 Aggregator & Market Interface Module

The Aggregator and Market Interface Module [6] follows the implementation presented in the Greek pilot (see Section 3.2.3), supporting DR communications via OpenADR2.0 protocol with SCOM's DRMS, as well as interfacing with other iFA's components. It also provides an interface, enabling the communication of electricity and network tariffs.

The next functionalities shall be supported by the A&M Interface Module [6] for the purposes of the Slovenian pilot:

- FN-AM-05 – Communication of flexibility potential
- FN-AM-08 – Receiving flexibility signal
- FN-AM-09 – Communication of flexibility signal

- FN-AM-11 – Communication of network tariffs from external system
- FN-AM-12 – Access to network tariffs to end users
- FN-AM-13 – Communication of electricity tariffs from external system
- FN-AM-14 – Access to electricity tariffs to end users

3.3.8 User Interface App

The UI for the Slovenian pilot has been developed as a hybrid mobile app, based on the React Native framework[11]. In the third pilot phase, it supports a wide range of functionalities. More specifically, it should meet the following requirements [4] for the third phase of the project:

- FN-UI-01 – Operation mode customisation
- FN-UI-02 – User-defined time and operational constraints
- FN-UI-03 – End-user feedback
- FN-UI-04 – Optimisation policy selection
- FN-UI-05 – Automation level customization
- FN-UI-07 – Supported system interface languages
- FN-UI-08 – Provision of consent for the schedules of dispatchable assets
- FN-UI-09 – DR notification policy
- FN-UI-10 – Insights into sustainability metrics
- FN-UI-11 – Real-time energy data
- FN-UI-12 – Past energy data
- FN-UI-13 – DR reports
- FN-UI-14 – Insights into energy efficiency
- FN-UI-15 – Customised alerts
- FN-UI-16 – Energy advising service
- FN-UI-17 – Inspection of energy tariffs
- FN-UI-21 – DR event notification
- FN-UI-22 – Presentation of DR event history
- FN-UI-27 – Actual schedules of assets
- FN-UI-28 – Estimation of energy costs
- FN-UI-29 – Benefits from iFA's operation
- NF-UI-03 – App walkthrough

Napaka! Vira sklicevanja ni bilo mogoče najti. shows the screen of the application that presents the daily e lectricity tariffs' fluctuation to the users. More screens of the application can be seen in D3.6 [15].

Figure 21: "My Tariffs" screen of the mobile application

[11] https://reactnative.dev/

4 Conclusion

Deliverable 6.7 is the commandment document, which is oriented towards prototype implementation and deployment in the third piloting phase, which includes Finland, Greece and Slovenia pilot partners.

The third phase of the Finnish pilot focuses on deploying the iFLEX Assistant (iFA) in both an apartment building and a supermarket. It consists of the following functional components: Resident Interface (only deployed in the apartment building), Facility Manager Interface, RAI, AFM, DTR, and WSI. The apartment building iFA oversees energy and flexibility management for the entire building, and it interacts with various external systems, including the Building Management System (BMS), Fingrid Open Data platform, and Finnish Meteorological Institute (FMI) weather forecasts, to efficiently monitor and control energy consumption. Similarly, the supermarket iFA exchanges information with its building automation system, optimizing energy usage and contributing to grid stability. Both deployments are integrated with Enerim's Aggregation Platform, facilitating interaction with the Nord Pool intraday market.

Greek pilot introduces an updated version of the iFLEX Assistant (iFA) detailing components, external systems, and communication protocols. The iFA components include the User Interface (UI), Aggregator and Market Interface (A&M), Resource Abstraction Interface (RAI), Digital Twin (DT), Automated Flexibility Management (AFM), and Security and Privacy modules. Additionally, the iFA interacts with external components such as ICOM's Demand Response Management System (DRMS), HERON's Real-time Energy Metering & Actuation Platform (REMAP), mobile app interacting with iFA through Push Notification Server, and other services including a Weather Service, HERON's tariff API, and a CO_2 emissions Service. The User Interface module offers a comprehensive set of functionalities tailored to the needs of pilot users, ranging from energy monitoring to demand response services. Similarly, the Aggregator & Market Interface module facilitates communication with external systems, enabling flexibility management and tariff information exchange. ICOM's Demand Response Management System (DRMS) plays a crucial role in orchestrating demand response actions, supporting functions such as flexibility potential assessment, signal exchange, and participation metrics calculation. HERON's Real-time Energy Metering & Actuation Platform (REMAP) provides a robust infrastructure for energy monitoring and management, enhanced with smart asset integration and a dedicated mobile application for user interaction.

The Slovenian iFLEX Assistant framework prototype in the third piloting phase encompasses several functional components aimed at enhancing energy management and flexibility within households. These components include the Enrolment Module, Home Energy Management System (HEMS), Resource Abstraction Interface (RAI), Smart Metering and Weather Data Ingest Module, Digital Twin, Demand Response Module, Trust Management Module, Security and Privacy Interface, and End-User Interface. Throughout the third phase of the Slovenian pilot, the focus was on the deployment, testing, and evaluation of the HEMS in working conditions, along with the integration validation of input data sources and improvements to the iFLEX Assistant end-user interface design. The HEMS functionalities and limitations include communication with various devices such as heat pumps, domestic hot water heaters, solar power plant inverters, and distribution electricity meters. Integration with the RAI is achieved via MQTT protocol, supporting messages for device configuration, state, and measurements.

5 List of figures and tables

5.1 Tables

6 References

[1] “D2.1 - Use cases and Requirements,” 2021.
[2] “D2.3 - Common architectureof iFLEX Framework,” 2021.
[3] J. R. T. K. J. K. Jari Palosaari, “MakingCity deliverable. D2.8 – Adaptation of Oulu ICT,” 2021.
[4] iFLEX project, “Revised Natural User Interfaces. iFLEX deliverable D3.5.,” 2022.
[5] IEC, “IEC 62746-10-1Systems interface between customer energy management system and the power management system - Part 10-1: Open Automated Demand Response (OpenADR)”.
[6] iFLEX project, “Revised Market and Aggregation Interface Module. iFLEX deliverable D4.5.,” 2022.
[7] “D3.9 Final automated flexibility management module,” 2023.
[8] “Revised Pilot Specifications. The IFLEX deliverable D7.2.,” 2022.
[9] “Final Pilot Specification. iFLEX deliverable D7.3.,” 2023.
[10] “Final architecture of iFLEX Framework. iFLEX deliverable D2.5.,” 2023.
[11] “Revised secure consumer data management module. IFLEX deliverable D4.8.,” 2022.
[12] “Final Resource abstraction interface. iFLEX deliverable D4.3.,” 2023.
[13] “Final hybrid-modelling module. iFlex deliverable D3.3.,” 2023.
[14] “D4.2 Revised Resouce Abstraction Interface,” 2022.
[15] “D3.6 - Final Natural User Interfaces,” 2023.